Gestion des données du web et interopérabilité

Béatrice Bouchou-Markhoff

UNIVERSITÉ FRANÇOIS - RABELAIS DE TOURS

Année Universitaire : 2012-2013

HABILITATION À DIRIGER DES RECHERCHES

Gestion des données du web et intéropérabilité

Discipline : Informatique

présentée et soutenue publiquement par :

Béatrice Bouchou Markhoff

le : 15/11/2013

JURY :
(Par ordre alphabétique)

Prénom	Nom	Grade	Établissement d'exercice
Véronique	Benzaken	Professeur des universités	Université Paris Sud
Mirian	Halfeld Ferrari Alves	Professeur des universités	Université d'Orléans
Denis	Maurel	Professeur des universités	Université François Rabelais Tours
Chantal	Reynaud	Professeur des universités	Université Paris Sud
Laurent	Romary	Directeur de recherche	INRIA Saclay - Ile de France

Remerciements

J'exprime toute ma gratitude aux membres du jury qui ont fait de la soutenance un moment d'ouvertures riches de potentiels. Les rapporteurs avaient déjà fait preuve dans leur lecture de mon mémoire d'une attention qui m'honore. Je remercie Véronique Benzaken pour ses appréciations chaleureuses et ses exhortations à sortir plus encore des cadres pour aborder tous les potentiels des données du web. Je remercie Laurent Romary pour avoir apporté l'éclairage de sa grande expérience de projets dans différents domaines de sciences humaines et sociales, en lexicographie comme en histoire. Je ne manquerai pas d'approfondir ses très nombreuses suggestions. Je remercie Chantal Reynaud pour ses remarques et suggestions très pertinentes sur les orientations les plus récentes de mes travaux, qu'elle avait déjà soigneusement évaluées en rapportant sur la thèse de Cheikh Niang, nous faisant ainsi bénéficier de son expertise dans le domaine de l'ingénierie des connaissances. Je dis encore et encore "mille merci" à Mirian Halfeld Ferrari Alves pour "les années XML" que nous avons partagées, si dynamiques qu'elles se sont prolongées malgré l'éloignement géographique. Qu'elles continuent à nous maintenir en collaboration longtemps. Enfin envers Denis Maurel, pour le naturel et la simplicité de son accueil lorsque j'ai décidé de reprendre une activité de recherche, pour sa constante sollicitude à mon égard, jusqu'à sa conduite de ce jury, je suis très reconnaissante.

Je remercie sincèrement tous ceux avec qui j'ai plaisir à travailler à Blois, l'équipe de recherche, les équipes administrative et technique des deux composantes, toute l'équipe pédagogique du DI de la fac de sciences et celle de la LP QSSI. Bien travailler ensemble crée naturellement des amitiés, chacun s'y reconnaîtra. J'ai aussi une pensée pour mes collaborateurs tourangeaux (du LI, du CESR, de l'IRHT, de CITERES, de l'UFR Sciences et Techniques...) bien sûr. Et au-delà, bien au-delà.

Au-delà du cadre du travail aussi, j'adresse un remerciement particulier à ceux qui m'accompagnent dans la vie, à nos enfants, à mon amour.

Table des matières

Liste des tableaux

Liste des figures

Introduction

Il y a plus d'une dizaine d'années qu'avec Mirian Halfeld Ferrari Alves j'ai initié un thème de recherche au sein de l'équipe "Bases de données et traitement automatique des langues naturelles" (BdTln) du Laboratoire d'Informatique de Tours : *les données du web*, incarnées au départ par XML et maintenant également par RDF et ses schémas (ontologies). Dans ce document je rassemble mes contributions à ce thème toujours actif, organisées selon les problématiques fondamentales en bases de données : la structure, les contraintes d'intégrité, la conception, la sémantique. Je le fais en suivant le fil conducteur qui a motivé mon intérêt pour XML d'abord, pour la démarche de définition de normes de représentation de ressources lexicales ensuite et pour les ontologies "légères" et leur intégration enfin : l'amélioration de *l'interopérabilité*, qui motive chaque avancée du web, augmentant à chaque étape son potentiel d'échanges et de collaboration dans l'élaboration de la connaissance.

Ce fil conducteur m'a d'abord amenée à travailler avec Mirian Halfeld Ferrari Alves sur XML et ses schémas : XML a été défini comme format d'échange, le fait de pouvoir définir le type des données échangées apportant l'interopérabilité. Le type des données XML est la structure que peuvent avoir les documents (schéma). La vérification de la validité d'un document XML par rapport à un type est souvent un préalable à l'exploitation du document et cette question ne prend réellement son sens que re-située dans le contexte du web : un environnement en évolution permanente. Ce fut l'une de nos premières propositions (DBPL 2003 [BAM03]), une revalidation incrémentale lors des mises à jour des documents. Nous avons ensuite considéré la question du maintien de la validité eut égard à l'évolution constante du web, soit par adaptation des contraintes de structure d'une part (thèse de Denio Duarte [Dua05] financée dans le cadre d'un projet CAPES-Cofecub, [BDAM09]), soit, d'autre part, par adaptation des mises à jour appliquées sur le document (thèse d'Ahmed Cheriat [Che06], [BCF+07]). En collaboration avec Agata Savary dans le cadre du projet ANR CODEX qui a financé Joshua Amavi comme ingénieur d'étude, ce même objectif d'adaptation a mené à une proposition complète de correction de documents XML par rapport à un ensemble de contraintes de structure [ABS13].

Du point de vue des bases de données, un schéma consiste en un ensemble de *contraintes de structure* d'une part et d'autre part en un ensemble de *contraintes d'intégrité* sur les données. Ces contraintes représentent des propriétés qui doivent être satisfaites par toutes les instances du schéma. Elles expriment une part de *sémantique* des données. Pouvoir maintenir automatiquement ces contraintes au fil de l'évolution du contenu de la base permet de garantir *l'intégrité des données*, dimension importante de la notion plus générale de qualité des données. Les contraintes d'intégrité les plus répandues sont les clés et clés étrangères, cas particuliers d'une classe de contraintes appelées *dépendances*. Une autre partie de

mes travaux en commun avec Mirian Halfeld Ferrari Alves concerne la définition et la vérification de contraintes de dépendances dans les documents XML (WebDB 2003 [BAM03], XSym 2004 [ABH+04]). Là encore, le contexte du web (en constante évolution) a été pris en compte dans nos propositions de vérification incrémentale de ces contraintes lors de mises-à-jour des documents (thèse de Maria Adriana Vidigal de Lima [Lim07] également financée dans le cadre du projet CAPES-Cofecub, [BAL09, BFL12]).

C'est avec ce point de vue sur les schémas et contraintes que j'ai abordé la question de la *conception* de schéma XML, avec pour cas d'étude une ressource lexicale élaborée sous la direction de Denis Maurel au sein de l'équipe BdTln : Prolexbase [Mau08]. Ce cas pratique est passionnant autant par le sujet, les noms propres et leurs relations dans les langues naturelles, que par son domaine plus général : la constitution de ressources lexicales *multilingues* pour le traitement automatique des langues. Depuis les années 90 ce domaine a suscité de grands projets internationaux, au fil desquels une démarche active de standardisation s'est constituée pour construire une série de propositions pour améliorer *l'interopérabilité* des ressources langagières, et ceci en tenant compte de la multiplicité des langues [FBG+09]. Après avoir conçu un premier schéma propre à Prolexbase [BTM05], nous avons suivi ces propositions de standards pour rendre la ressource plus partageable [BM08], plus évolutive également [MB13]. La démarche de standardisation, vue sous l'angle de la mise en commun des expériences ("best practices") est fondamentale pour l'interopérabilité.

L'idée de standard de représentation de ressources langagières amène à un autre domaine classique de la gestion des données, complémentaire aux bases de données : la *représentation des connaissances*. Il y a dans la notion même d'ontologie ([Gru09]) le principe de recherche de consensus, qui est également à la base d'un processus d'élaboration d'un standard de représentation. Ce dernier décrit en langage naturel le consensus, quand l'ontologie le formalise. L'intérêt de la formalisation est dans le potentiel de raisonnement automatique. Le lien d'origine entre bases de données et représentation des connaissances s'est renouvelé ces dernières années sous l'impulsion des technologies du web. Alors que les bases de données visent l'efficacité et la sûreté des *traitements automatiques* pour le stockage, les requêtes et les mises à jour des données, en s'appuyant sur un modèle mathématique efficace (modèle relationnel [AHV95] ou grammaires et automates d'arbres pour XML [BMW01, MLM01]), la représentation des connaissances, quant à elle, permet *aux utilisateurs* qui ont à modéliser et utiliser des informations, d'être efficaces dans ces démarches. Dans [CGL+11] en particulier, il est montré comment les ontologies peuvent servir d'interfaces entre les utilisateurs et les bases de données.

Des modèles de représentation des connaissances comme les graphes conceptuels [MC96, CM08] et les logiques de descriptions [BCM+03] font désormais partie du web, permettant d'exprimer ses ontologies, éléments clés de son niveau sémantique. Ces ontologies sont support d'interopérabilité, cette fois au niveau du sens, quand XML l'est au niveau de la syntaxe. L'émergence des *données liées*[1] les promeut au rôle de pivots pour l'interrogation de ces masses de données, également pour leur combinaison et leur *intégration*. L'intégration au niveau sémantique est précisément le thème central de la thèse de Cheikh Niang [Nia13] financée par l'AUF en co-tutelle avec l'Université Gaston Berger de Saint Louis du Sénégal, [NBLS11b, NBSL13].

1. http ://linkeddata.org/

Ce mémoire d'HDR résume donc la majeure partie de mes travaux menés entre 2002 et 2013 au sein du Laboratoire d'Informatique (EA 6300) de l'Université François Rabelais Tours, à savoir ceux qui concernent la gestion des données du web et l'interopérabilité des représentations, syntaxiques et sémantiques. Il est structuré en trois chapitres qui couvrent trois points de vue sur les données, ici données du web : leur validité par rapport à des contraintes de structure et des contraintes d'intégrité d'abord, la démarche de conception ensuite, leur sémantique enfin.

Le chapitre 1 porte essentiellement sur deux contributions parmi celles faites pour la gestion des données XML : d'une part l'algorithme de correction de document et d'autre part le cadre générique de définition et de validation de contraintes d'intégrité. L'un et l'autre permettent de tendre vers un fonctionnement maîtrisé des systèmes qui s'appuient sur XML, en nécessité permanente d'adaptation due à des situations d'échange, d'intégration, de réutilisation, etc. : mes perspectives immédiates dans ce cadre sont tournées vers l'extension et l'approfondissement des contributions présentées.

Le chapitre 2 présente mon expérience de conception de schéma XML pour la représentation d'une ressource lexicale multilingue de noms propres. Le cas particulier de Prolexbase illustre l'importance et la pertinence des standards pour la constitution de ressources lexicales, définis par la communauté internationale des chercheurs de ce domaine. La représentation de Prolexbase dans ces standards est actuellement mise à la disposition du public en format XML [2] : mes perspectives à court terme sont d'intégrer maintenant cette ressource au niveau sémantique du web.

Le chapitre 3 porte sur mes premières contributions au domaine de l'intégration sémantique de données. J'y résume la thèse de Cheikh Niang, qui consiste en une démarche générique (dépendant d'une ontologie de référence) de construction quasi-automatique d'un système médiateur sémantique, ouvert et dynamique. Les perspectives directes sont nombreuses, allant d'améliorations de la première version du système à sa généralisation, pour l'appliquer en particulier dans plusieurs projets en collaboration avec des équipes d'historiens de l'Université François-Rabelais de Tours.

En conclusion de ce mémoire, je présente mes perspectives de recherche à court et moyen terme, après avoir résumé mon parcours et le reste de mes travaux.

2. http ://www.cnrtl.fr/lexiques/prolex/

Chapitre 1

Schémas et contraintes d'intégrité pour XML

XML est une des briques de base du web, avec le codage Unicode et le protocole HTTP. Il a été défini pour être un format d'échange entre les applications, échanges de *documents* (de nombreux aspects sont dérivés de SGML) et de *données* (il est une incarnation des travaux des années 90 sur les données semi-structurée [ABS00]). Une grande partie de mes travaux porte sur des problématiques orientées bases de données concernant XML : le rapport entre les données et la structure que doit respecter leur représentation, l'évolution de ces données et de cette structure, ainsi que le respect par ces données de contraintes d'intégrité, avec la maintenance incrémentale de leur validité au fur et à mesure des évolutions.

Dans cette première partie je présente deux contributions dans ce cadre, la première sur le rapprochement entre un document et un schéma et la deuxième sur la définition et la validation de contraintes d'intégrité pour XML.

1.1 Document, schéma et évolution

Pour être un document XML, un fichier texte codé dans un format Unicode doit être composé d'un ensemble d'informations, certaines obligatoires, d'autres optionnelles et d'autres encore pouvant ne pas apparaître explicitement dans le fichier mais qui existent pour les applications qui exploitent le fichier, avec les valeurs par défaut qui ont été fixées par le W3C [1]. L'ensemble de ces informations est organisé en une arborescence appelée *infoset*. Le composant essentiel de cette arborescence, qui doit être décrit explicitement dans le fichier pour que ce dernier soit "un document XML", est ce que tout au long de ce document j'appellerai *l'arbre XML* : l'arbre des éléments XML, composé d'une part des éléments et noms d'attributs qui forment la *structure* et d'autre part des *valeurs*, ou données contenues dans le document, situées aux feuilles de l'arbre.

XML a été défini avec deux concepts différents pour déterminer la structure, il y a des éléments et des attributs, quand tout pourrait être élément. Les attributs peuvent

1. L'ensemble de ces informations est décrit dans la spécification du W3C : http ://www.w3.org/TR/2008/REC-xml-20081126/.

être utilisés pour associer à un élément un *ensemble* de propriétés dont les valeurs doivent être *atomiques* et *uniques* dans un élément donné. Les éléments quant à eux organisent la structure sous la forme d'un *arbre d'arité variable*, *étiqueté* et *ordonné*. Contrairement aux arbres d'arité fixe pour lesquels un nombre fixe de fils est associé à chaque étiquette de nœud [CDG+02], dans un arbre d'arité variable le nombre fini de fils qu'un noeud peut avoir est variable.

Lorsqu'il s'agit d'analyser ou de vérifier *la structure*, il est admis de ne pas considérer explicitement les attributs dans les raisonnements, pour deux raisons : (i) ils ne sont pas directement dans l'arbre ordonné, ils doivent donc être traités à part et (ii) leur traitement ne présente aucune difficulté particulière, comme nous l'avons montré dans [BDAL03]. Il est également inutile de considérer les données s'il est question uniquement de structure : pour les questions de validité d'un document XML par rapport à un schéma on ne considère donc que l'*arbre XML des éléments*. Lorsqu'il s'agit par contre d'analyser ou de vérifier des contraintes portant sur les *données* (contraintes d'intégrité) alors l'arbre XML admet une définition *étendue* pour représenter également les attributs et les valeurs que contient le document.

1.1.1 Documents et schémas

Tout au long des sections 1.1 et 1.2, axées sur la structure (représentée par le schéma), le document sera représenté par la définition suivante d'un arbre XML.

Définition 1.1.1 - *Arbre XML :* Un arbre XML est défini par son domaine, ensemble de **positions** $Pos(t)$ et une fonction t de $Pos(t)$ vers un **alphabet** Σ, lequel représente l'ensemble des noms d'éléments présents dans le document XML. Pour $v{\in}Pos(t)$, $t(v)$ est l'étiquette du nœud de l'arbre t en position v. Les positions sont des séquences d'entiers positifs. La séquence vide est dénotée par ϵ et elle correspond à la **position racine** de l'arbre. Le caractère "." dénote la **concaténation**. L'ensemble $Pos(t)$ est fermé par préfixe [2] et pour chaque position p de $Pos(t)$ toutes les positions à gauche de p, filles du même noeud père que p, existent également dans $Pos(t)$: $\forall_{i,j\in\mathbb{N}}\forall_{u\in\mathbb{N}^*}[[0{\leq}i{\leq}j, u.j{\in}Pos(t)] \Rightarrow u.i{\in}Pos(t)]$. L'ensemble des **feuilles** de t est défini par : $leaves(t) = \{u{\in}Pos(t) \mid \nexists_{i\in\mathbb{N}}\ u.i{\in}Pos(t)\}$.
La **taille** de t (nombre de positions dans$Pos(t)$) est notée $|t|$, le nombre de fils directs du nœud racine de t est noté $\bar{t}$, la séquence des étiquettes des fils directs du nœud racine de t est noté $s(\bar{t})$ et la hauteur de t est notée d_t. Un arbre **vide** (pour lequel $Pos(t_\emptyset) = \emptyset$) est dénoté $t_\emptyset$.

Exemple 1.1.1 *Le document XML suivant (dont j'ai omis les valeurs contenues dans les éléments) correspond à l'arbre XML t représenté en figure 1.1 :*

```
<root>
   <a> <c> </c> <d> </d> </a>
   <b> <c> </c> </b>     <b> <c> </c> </b>
</root>
```

2. La relation préfixe sur $\mathbb{N}^*$, dénotée par $\preceq$ est définie ainsi : $u \preceq v$ si et seulement si il existe un $w \in \mathbb{N}^*$ tel que $u.w = v$. La séquence u est un préfixe propre de v, noté $u \prec v$, si et seulement si $w \neq \epsilon$. Un ensemble $Pos(t) \subseteq \mathbb{N}^*$ est fermé par préfixe si ($u \preceq v$ et $v \in Pos(t)$) implique que $u \in Pos(t)$.

Dans l'arbre XML t de la figure 1.1 on a :
- $\Sigma = \{root, a, b, c, d\}$
- $Pos(t) = \{\epsilon, 0, 0.0, 0.1, 1, 1.0, 2, 2.0\}$
- $t = \{(\epsilon, root), (0, a), (0.0, c), (0.1, d), (1, b), (1.0, c), (2, b), (2.0, c)\}$
- $t(\epsilon) = root, t(0) = a, t(0.0) = c,$ *etc.*
- $leaves(t) = \{0.0, 0.1, 1.0, 2.0\}$
- $|t| = 8$
- $\bar{t} = 3$
- $s(\bar{t}) = a.b.b$
- $d_t = 2$ □

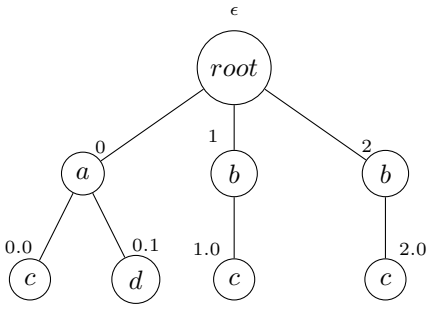


Figure 1.1 – Arbre XML du document de l'exemple 1.1.1.

Définition 1.1.2 - *Sous-arbre et arbre partiel :* Etant donné un arbre XML non vide t, une position $p \in \mathbb{N}^*$ et $i \in \mathbb{N}$ tel que $-1 \leq i \leq \bar{t} - 1$, les notions de sous-arbre et d'arbre partiel sont définies comme suit :
- $t_{|_p}$ dénote **le sous-arbre** de t dont la racine est en position $p \in Pos(t)$, précisément :
  1. Chaque nœud de t sous la position p appartient à $t_{|_p}$ $(\forall_{u \in \mathbb{N}^*}[[p.u \in Pos(t)] \Rightarrow [u \in Pos(t_{|_p}) \text{ et } t_{|_p}(u) = t(p.u)]])$.
  2. Il n'existe pas de nœud de $t_{|_p}$ qui n'appartienne pas à t $(\forall_{u \in Pos(t_{|_p})} p.u \in Pos(t))$.
- $t\langle i \rangle$ dénote **l'arbre partiel** de t qui contient la racine de t et $t_{|_0}$, $t_{|_1}$, ..., $t_{|_i}$, soit :
  1. Toutes les positions dans $t\langle i \rangle$ sont les mêmes que celles des premiers sous-arbres de t $(Pos(t\langle i \rangle) = \{v \in Pos(t) | v = \epsilon \text{ or } \exists_{0 \leq k \leq i} \exists_{u \in \mathbb{N}^*} v = k.u\})$.
  2. Toutes les étiquettes de $t\langle i \rangle$ sont celles de t $(\forall_{v \in Pos(t\langle i \rangle)} t\langle i \rangle(v) = t(v))$.

Tout sous-arbre ou arbre partiel d'un arbre XML est lui-même un arbre XML au sens de la définition 1.1.1 et pour tout arbre non vide t on a $t_{|_\epsilon} = t$, $t\langle \bar{t} - 1 \rangle = t$, et $t\langle -1 \rangle = \{(\epsilon, t(\epsilon))\}$.

Exemple 1.1.2 *La figure 1.2 montre le sous-arbre $t_{|_1}$ et l'arbre partiel $t\langle 1 \rangle$ de l'arbre t de la figure 1.1.* □

Les documents XML peuvent être associés à une définition de leur structure appelée *schéma*. C'est une caractéristique fondamentale pour l'interopérabilité puisque le schéma

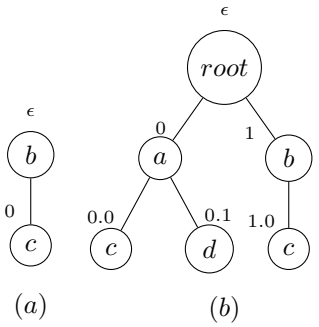


FIGURE 1.2 – (a) Sous-arbre $t_{|_1}$ et (b) arbre partiel $t\langle 1\rangle$ de l'arbre t de la fgure 1.1

indique à toute application comment exploiter le document. De nombreux langages pour exprimer un schéma ont été proposés, à l'heure actuelle trois sont très utilisés en pratique : le formalisme DTD (Document Type Definition) d'origine couplé à XML, le langage XML Schema [3] du W3C et le langage Relax-NG [4] de l'OASIS. Quel que soit le langage utilisé, la définition de la structure d'un document XML consiste en une *grammaire d'arbres* et, comme pour les langages de mots, vérifier si un arbre appartient au langage engendré par cette grammaire revient à vérifier si un *automate d'arbre* accepte cet arbre. Ainsi, définir la structure que des documents XML doivent respecter c'est définir un langage d'arbres.

Un ensemble de travaux [BW98b, BW98a, BMW01, MLM01, MLMK05] ont montré comment étendre aux arbres XML, d'arité variable, les formalisations, algorithmes et propriétés établis auparavant pour les arbres d'arité fixe [CDG+02] [5]. La définition suivante est celle des automates d'arbre ascendants adaptés aux arbres d'arité variable, qui peuvent être construits à partir d'une définition de schéma et dont l'exécution correspond donc au processus de validation d'un document XML par rapport à ce schéma.

Définition 1.1.3 - *Automate de schéma [BH03] :* Soit Σ l'alphabet des noms d'éléments et d'attributs définis dans un schéma D, l'automate d'arbre ascendant correspondant à D est un quadruplet $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$ dans lequel Q est l'ensemble des états, $Q_f \subseteq Q$ est l'ensemble des états finaux et Δ est l'ensemble des règles de transition de la forme $a, C, E \rightarrow q$, où *(i)* $a \in \Sigma$, *(ii)* C est un couple d'ensembles disjoints d'états $(S_{compulsory}, S_{optional})$, *(iii)* E est une expression régulière sur l'alphabet Q et *(iv)* $q \in Q$.

Il s'agit de la définition que nous avons utilisée dans nos propositions et qui prend en compte les attributs car nous avons montré comment traiter aussi les attributs lors des validations initiales et incrémentales [BH03, BCF+07]. Un *automate qui ne considère pas les attributs* est identique à celui défini ici, sans composant C dans les règles de transition.

3. http ://www.w3.org/XML/Schema
4. http ://relaxng.org/
5. Il est également toujours possible de transformer un arbre d'arité variable en arbre binaire, de différentes manières, et d'appliquer alors directement les connaissances établies sur les automates d'arbres binaires [Hos11].

Définition 1.1.4 - *Exécution d'un automate de schéma [BCF+07] :* L'exécution de l'automate $\mathcal{A}$ sur un arbre XML étendu[6] T est représentée par un arbre r tel que (i) $dom(r) = dom(T)$ et (ii) à chaque position p est affecté un ensemble d'états $\mathcal{Q}_p$. Le calcul de $\mathcal{Q}_p$ revient à vérifier si les contraintes sur les attributs et les sous-éléments de l'élément en position p sont respectées. Pour cela, l'ensemble $\mathcal{Q}_p$ reçoit un état q si :

1. Il existe une règle de transition $a, (S_{compulsory}, S_{optional}), E \rightarrow q$ pour $a = T(p)$.
2. $S_{compulsory} \subseteq \mathcal{Q}_{att}$ et $\mathcal{Q}_{att} \backslash S_{compulsory} \subseteq S_{optional}$ où $\mathcal{Q}_{att}$ est l'ensemble des états associés aux attributs trouvés dans T sous la position p.
3. Il existe un mot $w = q_i, \ldots, q_n$ in $L(E)$ tel que $q_i \in \mathcal{Q}_{p.i}, \ldots, q_n \in \mathcal{Q}_{P.n}$, où $Q_{p.i}$ désigne l'ensemble d'états associés à la position fille de p contenant le premier sous-élément fils[7] et $Q_{p.n}$ désigne l'ensemble d'états associés à la position fille de p contenant le dernier sous-élément fils. En d'autres termes il faut tester si $L(E_{aux}) \cap L(E) \neq \emptyset$[8], où E_{aux} est l'expression régulière représentant les mots que l'on peut construire à partir des ensembles d'états des sous-éléments de p, *i.e.*, $E_{aux} = (q_1^0 \mid q_1^1 \mid \ldots \mid q_1^{k_1}) \ldots (q_n^0 \mid q_n^1 \mid \ldots \mid q_n^{k_n})$. Ce test correspond au calcul de l'intersection des automates à états finis correspondant aux expressions E_{aux} et E, connu pour être en temps $O(n^2)$ où n est la plus grande des tailles des deux automates [HMU01, KLV00].

Une exécution r de $\mathcal{A}$ sur T est *réussie* si $r(\epsilon)$ contient au moins un état final. L'arbre XML étendu T est *valide* par rapport au schéma D représenté par l'automate $\mathcal{A}$ si l'exécution r de $\mathcal{A}$ sur T est réussie.

La complexité du calcul d'une exécution r de $\mathcal{A}$ sur T est linéaire sur la taille de T et dépend par ailleurs de la forme de l'automate $\mathcal{A}$. En effet, un automate $\mathcal{A}$ et son exécution ont des caractéristiques différentes selon le langage utilisé pour définir le schéma, qui correspond à l'un ou l'autre des trois types de grammaires régulières d'arbres discutés dans [MLMK05], dont je rappelle les principes dans la définition suivante :

Définition 1.1.5 - *Classes de langages réguliers d'arbres [MLMK05] :* dans un automate $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$ deux états q_1 et q_2 sont *en compétition* si Δ contient différentes règles de transition, par exemple $a, S_1, E_1 \rightarrow q_1$ et $a, S_2, E_2 \rightarrow q_2$ pour le même symbole a (avec des états q_1 et q_2 différents). A partir de cette notion, les langages réguliers d'arbres se répartissent en trois classes selon la hiérarchie LTL $\subset$ STTL $\subset$ RTL, en d'autres termes la classe RTL est la plus expressive. Les caractéristiques de chacune de ces trois classes sont les suivantes :

- *RTL* (*Regular Tree Languages*) – les *langages d'arbres réguliers*, sont reconnus par tout automate correspondant à la définition 1.1.3, sans restriction particulière sur les règles de transition.
- *LTL* (*Local Tree Languages*) – les *langages d'arbres locaux* sont reconnus par tout automate correspondant à la définition 1.1.3 dont l'ensemble Δ *ne contient aucun état en compétition*. En d'autres termes il n'y a jamais deux règles de transition pour un même symbole de Σ.

6. Un arbre XML étendu contient aussi attributs et données, comme précisé dans la définition 1.3.1.
7. Les attributs sont les premier fils de p dans un arbre XML étendu.
8. $L(E)$ est le langage engendré par l'expression régulière E.

– *STTL* (*Single-Type Tree Language*) – les *langages d'arbres à type unique* sont reconnus par tout automate correspondant à la définition 1.1.3 pour lesquels (i) aucune règle de transition ne contient deux états en compétition et (ii) l'ensemble Q_f est un singleton. En d'autres termes, même s'il est possible d'avoir des éléments de même nom avec des structures différentes (états en compétition), ces définitions concurrentes apparaissent dans des contextes disjoints (jamais dans la même règle de transition) et au final un seul état détermine la validité, ou pas, de l'ensemble du document.

Pour la classe *RTL*, l'exécution de l'automate nécessite une gestion effective d'un ensemble d'états à chaque nœud de l'arbre à valider ainsi que le calcul de l'intersection de deux langages de mots d'états à chaque nœud également. Les structures spécifiées en *specialized DTD* [Via01, BPV04] ou en *Relax-NG* peuvent définir des langages réguliers d'arbres. Pour la classe *LTL*, l'exécution de l'automate est extrêmement simple, il n'y a qu'un état possible pour chaque nœud et il suffit de vérifier si le mot formé par les états associés aux sous-éléments du nœud courant est dans le langage de l'expression régulière. Les structures spécifiées en *DTD* ne peuvent définir que des langages d'arbres locaux. Pour la classe *STTL*, l'exécution de l'automate n'a à considérer qu'un seul état à chaque nœud car le contexte détermine la règle de transition à appliquer. Les structures spécifiées en *XML Schema* peuvent définir des langages locaux d'arbres ou des langages d'arbres à type unique.

Exemple 1.1.3 Soit $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$ un automate dans lequel $\{a, b, c\} \subset \Sigma$, $\{q_1, q_2, q_3, q_{a1}, q_{a2}, q_b, q_c\} \subset Q$ et $Q_f = \{q_c\}$. Soient les règles de transition suivantes[9], où q_{a1} et q_{a2} sont des états en compétition et q_1, q_2 et q_3 correspondent à des règles de transition simples, sans état en compétition donc sans les états q_{a1} et q_{a2} :

$$
\begin{array}{lll}
(1) & a, (\emptyset, \emptyset), q_1.q_2? & \rightarrow q_{a1} \\
(2) & a, (\emptyset, \emptyset), q_1.q_3? & \rightarrow q_{a2} \\
(3) & b, (\emptyset, \emptyset), (q_{a1} + q_{a2})^* & \rightarrow q_b \\
(3') & b, (\emptyset, \emptyset), (q_{a1})^* & \rightarrow q_b \\
(4) & c, (\emptyset, \emptyset), (q_b)^+ & \rightarrow q_c \\
(4') & c, (\emptyset, \emptyset), (q_{a2})^+ & \rightarrow q_c
\end{array}
$$

Si Δ contient les règles (1), (2), (3) et (4), alors le langage reconnu par $\mathcal{A}$ est un langage d'arbres régulier qui n'est pas un langage d'arbres à type unique. Si Δ contient les règles (1), (2), (3′) et (4′) alors le langage reconnu par $\mathcal{A}$ est un langage d'arbres à type unique qui n'est pas un langage d'arbres local. Si Δ contient les règles (1), (3′) et (4) alors le langage reconnu par $\mathcal{A}$ est un langage d'arbres local. □

A partir du cadre défini par l'ensemble de définitions que je viens de rappeler, nous avons montré dans [BDAL03, BH03, BDA+04, BCHFAS06b, BCHFAS06a, BCF+07, BD07, BDAM09] comment :

– (*i*) Valider des documents par rapport à un schéma quelconque avec un automate de schéma, donc en considérant les attributs et, dans le cas DTD, en vérifiant également les contraintes ID/IDREF(S) [BDAL03, BCF+07].

9. Etant donné un alphabet $Q = \{q_1, ..., q_n\}$, l'ensemble des expressions régulières sur Q est inductivement défini par : $E ::= \emptyset \mid \epsilon \mid q_i \mid E + E \mid E.E \mid E^+ \mid E^* \mid E? \mid (E)$.

- (*ii*) Assurer une validation *incrémentale* en cas de mise-à-jour du document, c'est-à-dire en limitant les vérifications aux voisinages des nœuds mis-à-jour. Dans le cas DTD c'est très simple, dans le cas XML Schema cela reste limité en terme de complexité et dans le cas des langages d'arbres régulier le calcul est en $O(m.n^2.h)$, où m est le nombre d'opérations de mise-à-jour effectuées, n est le plus grand nombre de sous-éléments d'un même nœud dans l'arbre XML et h est la hauteur de l'arbre XML [BH03, BCF+07] (thèse d'Ahmed Cheriat [Che06]).
- (*iii*) En cas d'invalidité entraînée par une mise-à-jour proposer des *corrections du document mis à jour* pour rétablir la validité [BCHFAS06b, BCHFAS06a] (thèse d'Ahmed Cheriat [Che06]),
- (*iv*) Toujours en cas d'invalidité entraînée par une mise-à-jour, proposer des *évolutions conservatives du schéma* pour rétablir la validité du document mis-à-jour tout en maintenant également celle des autres documents précédemment valides [BDA+04, BDAM09], ceci en introduisant les changements nécessaires pour admettre la mise-à-jour au sein du processus de réduction d'un automate de Glushkov en expression régulière proposé par [CZ00] (thèse de Denio Duarte [Dua05]).
- (*v*) Assister les administrateurs de bases de données XML dans leurs mises-à-jour d'un schéma pour lequel ils ont des documents valides et qui doivent le rester [BD07].

Mais dans ce mémoire je ne donnerai que les grandes lignes de mon travail le plus récent sur la question des schémas : la correction de la structure d'un document par rapport à un schéma de la classe *LTL* ou *STTL* [ABS13]. Dans ce cadre, les arbres sont des arbres XML (définition 1.1.1) et l'automate de schéma est simplifié : il ne considère pas les attributs (pas de composant C dans les règles de transition de la définition 1.1.3 et pas d'étape 2 dans l'exécution décrite en définition 1.1.4). Dans la suite, j'appellerai l'automate de schéma simplifié un *automate de schéma-élément* et j'utiliserai les définitions suivantes.

Définition 1.1.6 - *Arbre valide :* étant donné un schéma D représenté par un automate de schéma-élément $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$, un arbre XML t est **valide** par rapport au schéma D si et seulement si l'exécution r de $\mathcal{A}$ sur t est réussie, c'est-à-dire $r(\epsilon) \subset Q_f$.

Définition 1.1.7 - *Arbre localement valide :* étant donné un schéma D représenté par un automate de schéma-élément $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$, un arbre XML t est **localement valide** par rapport au schéma D si et seulement si l'exécution r de $\mathcal{A}$ sur t est telle que $r(\epsilon) \subset Q$.

Définition 1.1.8 - *Arbre partiellement valide :* étant donné un schéma D représenté par un automate de schéma-élément $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$, un arbre XML t est **partiellement valide** par rapport au schéma D si et seulement si t est un arbre partiel d'un arbre valide par rapport à D.

1.1.2 Evolution des documents

On a à faire évoluer les documents XML dans de nombreuses situations, c'est en général parce que les données évoluent, par exemple on doit insérer de nouveaux patients, modifier des résultats d'expériences, supprimer des stocks les produits vendus, etc. mais cela peut

également être parce que le schéma évolue avec l'évolution des activités (intégration de nouveaux paramètres) et des réseaux de coopération (intégration de données et intégration de schémas). Très nombreuses également sont les situations où l'on a à travailler sur des documents XML qui sont "proches" d'un document donné. Dans [TCTT11] les auteurs citent la recherche et la composition automatique de services web, le calcul consistant des réponses aux requêtes sur des données incomplètes ou inconsistantes [SC06, TVY08], la classification automatique de documents XML [BGM04, XMXV06, BGM08], etc.

Les modifications opérées dans un arbre, qui produisent un nouvel arbre, sont appelées "opérations d'édition". On peut se référer à [Bar95] pour un panorama sur les différentes opérations d'édition dans les arbres et les différents algorithmes de transformation d'un arbre en un autre arbre. Les opérations d'édition les plus élémentaires ne modifient qu'un seul nœud de l'arbre, en ce sens elles sont appelées *opérations d'édition-nœud* : changement de l'étiquette d'un nœud, insertion d'un nouveau nœud et suppression d'un nœud. Dans un arbre XML, une opération d'édition-nœud est applicable à une position p à condition de vérifier certaines conditions. Par exemple une insertion d'un nœud dans l'arbre de la figure 1.1 est possible en toute position sauf ϵ, mais également à certaines positions inexistantes, par exemple 2.1, de plus une insertion en une position existante va provoquer un décalage des positions à droite, etc. Pour préciser les opérations d'édition-nœud nous utilisons les définitions suivantes :

Définition 1.1.9 - *Ensembles caractéristiques de positions dans un arbre XML :* Soient t un arbre XML, p une position de t telle que $p = \epsilon$ ou $p = u.i$ ($u \in \mathbb{N}^*$ et $i \in \mathbb{N}$).

- La **frontière d'insertion** de t, notée $InsFr(t)$, est l'ensemble des positions qui ne sont pas dans t dans lesquelles il est possible d'opérer une insertion. Précisément :
  1. $InsFr(t_\emptyset) = \{\epsilon\}$.
  2. Si $t \neq t_\emptyset$ alors $InsFr(t) = \{v.j \notin Pos(t) \mid v \in Pos(t)$ et $j \in \mathbb{N}$ et $[(j = 0)$ ou $((j \neq 0)$ et $v.(j-1) \in Pos(t))]\}$.
- L'ensemble des **positions à changer** par rapport à p, noté $ChangePos_p(t)$, est l'ensemble des positions qui doivent être soit supprimées, soit décalées à gauche, soit décalées à droite en cas d'insertion ou de suppression de nœud en p. Précisément :
  1. $ChangePos_\epsilon(t) = \{\epsilon\}$
  2. Si $p \neq \epsilon$ alors $ChangePos_p(t) = \{w \mid w \in Pos(t),\ w = u.k.u',\ i \leq k < \bar{t_u}$ et $u' \in \mathbb{N}^*\}$.
- L'ensemble des **positions décalées à droite** par rapport à p, noté $ShiftRightPos_p(t)$, est :
  1. $ShiftRightPos_\epsilon(t) = \emptyset$.
  2. Si $p \neq \epsilon$ alors $ShiftRightPos_p(t) = \{w \mid w = u.(k+1).u',\ u.k.u' \in Pos(t),\ i \leq k < \bar{t_{|u}}$ et $u' \in \mathbb{N}^*\}$.
- L'ensemble des **positions décalées à gauche** par rapport à p, noté$ShiftLeftPos_p(t)$, est :
  1. $ShiftLeftPos_\epsilon(t) = \emptyset$.
  2. Si $p \neq \epsilon$ alors $ShiftLeftPos_p(t) = \{w \mid w = u.(k-1).u',\ u.k.u' \in Pos(t),\ i+1 \leq k < \bar{t_{|u}}$ et $u' \in \mathbb{N}^*\}$.

Exemple 1.1.4 *Pour l'arbre t en figure 1.1 :*
- $InsFr(t) = \{0.0.0, 0.1.0, 0.2, 1.0.0, 1.1, 2.0.0, 2.1, 3\}$
- $ChangePos_1(t) = \{1, 1.0, 2, 2.0\}$
- $ShiftRightPos_1(t) = \{2, 2.0, 3, 3.0\}$
- $ShiftLeftPos_1(t) = \{1, 1.0\}$ □

Définition 1.1.10 - *Opérations d'édition-nœud :* Etant donnés un alphabet Σ et le caractère réservé $/ \notin \Sigma$, une **opération d'édition-nœud** ed est le triplet (op, p, l), où $op \in \{relabel, add, delete\}$, $p \in \mathbb{N}^*$ et $l \in \Sigma \cup \{/\}$. Soit t un arbre XML, l'opération d'édition-nœud ed est **définie sur** t si et seulement si l'une des conditions suivantes est satisfaite :
- $op = relabel$, $l \in \Sigma$ et $p \in Pos(t)$
- $op = add$, $l \in \Sigma$ et $p \in Pos(t) \setminus \{\epsilon\} \cup InsFr(t)$
- $op = delete$, $l = /$ et $p \in leaves(t)$

Soit une opération d'édition-nœud ed, une **dérivation** D_{ed} est une fonction partielle de tout arbre t sur lequel ed est définie vers un autre arbre t', notée $t \xrightarrow{D_{ed}} t'$ ou simplement $t \xrightarrow{ed} t'$, qui transforme t en t' selon les conditions suivantes :
- Pour une opération **relabel**, la dérivation remplace l'étiquette de la position p ; si $ed = (relabel, p, l)$ alors :
  1. $Pos(t') = Pos(t)$,
  2. $t'(p) = l$,
  3. $\forall_{p' \in Pos(t') \setminus \{p\}} t'(p') = t(p')$.
- Pour une opération **add** la dérivation insère un nœud en p en décalant vers la droite les positions à droite de p ; si $ed = (add, p, l)$ alors :
  1. $Pos(t') = (Pos(t) \setminus ChangePos_p(t)) \cup ShiftRightPos_p(t) \cup \{p\}$,
  2. $t'(p) = l$,
  3. $\forall_{p' \in (Pos(t) \setminus ChangePos_p(t))} t'(p') = t(p')$,
  4. $\forall_{p' \in ShiftRightPos_p(t)} [[p = u.i,\ p' = u.(k+1).u' \text{ et } i, k \in \mathbb{N},\ u, u' \in \mathbb{N}^*] \Rightarrow t'(p') = t(u.k.u')]$.
- Pour une opération **delete** la dérivation supprime une feuille en décalant vers la gauche les positions à droite de p ; si $ed = (delete, p, /)$ alors :
  1. $Pos(t') = (Pos(t) \setminus ChangePos_p(t)) \cup ShiftLeftPos_p(t)$,
  2. $\forall_{p' \in (Pos(t) \setminus ChangePos_p(t))} t'(p') = t(p')$,
  3. $\forall_{p' \in ShiftLeftPos_p(t)} [[p = u.i,\ p' = u.(k-1).u' \text{ et } i, k \in \mathbb{N},\ u, u' \in \mathbb{N}^*] \Rightarrow t'(p') = t(u.k.u')]$.

Exemple 1.1.5 *L'opération d'édition-nœud $ed = (add, 1, a)$ est définie sur l'arbre en figure 1.1 et la dérivation transforme t en t' dessiné en figure 1.3.* □

Définition 1.1.11 - *Séquence d'opérations d'édition-nœud :* Soient t un arbre XML, $0 \leq n$ et $ed_1, ed_2, \ldots ed_n$ des opérations d'édition-nœud. La **séquence d'opérations d'édition-nœud** $nos = \langle ed_1, ed_2, \ldots ed_n \rangle$ est **définie sur** t si et seulement si il existe une séquence d'arbres $t_0, t_1, \ldots, t_n$ telle que :

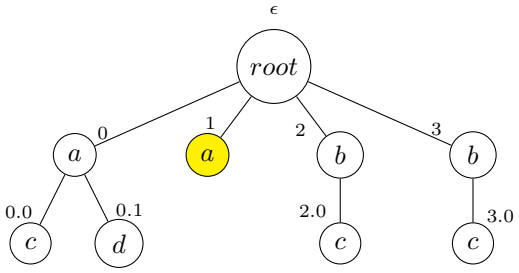


FIGURE 1.3 – Résultat de la dérivation pour $ed = (add, 1, a)$ sur l'arbre t en figure 1.1.

- $t_0 = t$
- $\forall_{0<k\leq n} ed_k$ est définie sur t_{k-1} et $t_{k-1} \xrightarrow{ed_k} t_k$.

Soit une séquence d'opérations d'édition-nœud nos, la **dérivation** D_{nos} est une fonction partielle de tout arbre t sur lequel nos est définie vers un autre arbre t', notée $t \xrightarrow{D_{nos}} t'$ ou simplement $t \xrightarrow{nos} t'$, qui transforme t en t' si et seulement si il existe une séquence d'arbres $t_0, t_1, \ldots, t_n$ comme défini auparavant et $t_n = t'$.
La **séquence vide d'opérations d'édition-nœud** est notée $nos_\emptyset$ et on a $t \xrightarrow{nos_\emptyset} t$.
Deux séquences d'opérations d'édition-nœud nos_1 et nos_2 sont dites **équivalentes** si et seulement si pour tout arbre t sur lequel elles sont définies la dérivation de t par nos_1 et la dérivation de t par nos_2 produisent le même arbre : $nos_1 \equiv nos_2$ si et seulement si :
$\forall_t [[nos_1$ et nos_2 sont définies sur t, $t \xrightarrow{nos_1} t_1$ et $t \xrightarrow{nos_2} t_2] \Rightarrow t_1 = t_2]$

Exemple 1.1.6 *La séquence d'opérations d'édition-nœud* $nos = \langle (relabel, 0.1, c),$ $(delete, 0.0, /), (relabel, 0, b)\rangle$ *est définie sur l'arbre* t *de la figure 1.1 et la dérivation par* nos *de* t *produit* t'_1 *dessiné en figure 1.8.*
A noter que : $nos \equiv \langle (delete, 0.1, /), (relabel, 0, b)\rangle \equiv \langle (relabel, 0, b), (delete, 0.1, /)\rangle$. □

Pour un arbre XML donné t et un ensemble de séquences d'opérations d'édition-nœud $NOS = \{nos_1, \ldots, nos_n\}$ définies sur t, la dérivation de t par NOS, c'est à dire par chaque $nos_i \in NOS$: $(t \xrightarrow{nos_i} t_i)$, est notée $t \xrightarrow{NOS} \{t_1, \ldots, t_n\}$. Enfin, des séquences d'opérations d'édition-nœud particulières correspondent à des opérations d'édition plus "haut niveau" d'ajout ou de retrait de sous-arbres entiers par exemple, telles les *opérations de mise-à-jour* couramment utilisées (cf. section 1.4.2). Elles sont précisées dans la définition suivante :

Définition 1.1.12 - *Opérations d'édition-arbre :* Soient l'alphabet Σ, une **opération d'édition-arbre** ted est un triplet (op, p, τ), où $op \in \{insert, remove, replace\}$, $p \in \mathbb{N}^*$ et τ est un arbre XML dont les étiquettes sont dans Σ. Pour un arbre XML t d'alphabet Σ l'opération d'édition-arbre ted est **définie sur** t si et seulement si l'une des conditions suivantes est vérifiée :
- $op = insert$ et $p \in Pos(t) \setminus \{\epsilon\} \cup InsFr(t)$
- $op = remove$, $p \in Pos(t)$ et $\tau = t_\emptyset$
- $op = replace$ et $p \in Pos(t)$

Soit une opération d'édition-arbre ted, la **dérivation** D_{ted} est une fonction partielle de tout arbre t sur lequel ted est définie, vers un arbre t', notée $t \xrightarrow{D_{ted}} t'$ ou simplement $t \xrightarrow{ted} t'$, qui transforme t en t' selon les conditions suivantes :

- Pour une opération **insert**, la dérivation insère un nouvel arbre en p, en décalant des positions vers la droite ; si $ted = (insert, p, \tau)$ alors :
  $t = t_0 \xrightarrow{(add,p.v_1,\tau(v_1))} t_1 \xrightarrow{(add,p.v_2,\tau(v_2))} t_2 \ldots \xrightarrow{(add,p.v_n,\tau(v_n))} t_n = t'$ où $v_1, \ldots, v_n$ sont les positions de τ visitées dans l'ordre préfixe.
- Pour une opération **remove**, la dérivation supprime le sous-arbre enraciné en p ; si $ted = (remove, p, t_\emptyset)$ alors :
  $t = t_0 \xrightarrow{(delete,p.v_1,/)} t_1 \xrightarrow{(delete,p.v_2,/)} t_2 \ldots \xrightarrow{(delete,p.v_n,/)} t_n = t'$ où $v_1, \ldots, v_n$ sont les positions du sous-arbre $t_{|_p}$ visités dans l'ordre postfixe inverse (de droite à gauche).
- Pour une opération **replace**, une dérivation possible est de supprimer le sous-arbre enraciné en p puis d'insérer τ à cette place, mais la dérivation de coût minimal est de *corriger* le sous-arbre en p vers le sous-arbre τ, en appliquant l'algorithme décrit en section 1.2.

Toute opération d'édition-arbre ted peut s'exprimer en terme de séquence d'opérations d'édition-nœud nos selon la définition 1.1.12. Nous disons que ted et nos sont **t-equivalentes**, noté $ted \equiv_t nos$. D'après la définition 1.1.12, à chaque opération d'édition-arbre correspond exactement une séquence d'opérations d'édition-nœud t-equivalente.

Exemple 1.1.7 L'opération d'édition-arbre $ted = (insert, 3, \tau_1)$ où $\tau_1 = \{(\epsilon, b), (0, c)\}$ est définie sur l'arbre t en figure 1.1 et elle est t-équivalente à $\langle (add, 3, b), (add, 3.0, c) \rangle$. □

Toute dérivation induit un coût de transformation non négatif. Il peut être fixé à 1 pour chaque opération d'édition-nœud, mais peut aussi varier selon les cas d'application [10].

Définition 1.1.13 - *Coût d'une séquence d'opérations d'édition-nœud :* Pour toute opération d'édition-nœud ed, un coût $cost(ed)$ est défini comme le coût non négatif de dérivation d'un arbre par ed. Soit la séquence d'opérations d'édition-nœud $nos = \langle ed_1, ed_2, \ldots, ed_n \rangle$, le coût de nos est défini par $Cost(nos) = \Sigma_{i=1}^{n}(cost(ed_i))$. Le coût d'une opération d'édition-arbre ted est égal au coût de la séquence nos t-équivalente, i.e. $Cost(ted) = Cost(nos)$ où $ted \equiv_t nos$. Soit l'ensemble de séquences d'opérations d'édition-nœud NOS, le coût minimum de NOS est défini par $MinCost(NOS) = \min_{nos \in NOS} \{Cost(nos)\}$.

Voici enfin la définition de la distance entre deux arbres et entre un arbre et un langage d'arbres, fondamentales pour l'algorithme de correction de document. On notera que la première correspond bien à la distance entre deux arbres proposée par [Sel77].

Définition 1.1.14 - *Distances d'arbres :* Soient t et t' des arbres et soit $NOS_{t \to t'}$ l'ensemble de toutes séquences d'opérations d'édition-nœud nos telles que $t \xrightarrow{nos} t'$. La distance entre les deux arbres t et t' est définie par $dist(t, t') = MinCost(NOS_{t \to t'})$. La distance entre un arbre t et un langage d'arbres L est définie par $DIST(t, L) = \min_{t' \in L}\{dist(t, t')\}$.

10. Dans le prototype développé pour l'algorithme de correction de document le coût de chaque opération est un paramètre.

1.2 Correction d'un document par rapport à un schéma

L'algorithme complet de correction d'un document par rapport à un schéma que nous avons publié dans [ABS13] est une contribution particulièrement importante pour *pouvoir garantir un fonctionnement maîtrisé* dans un contexte d'approximations très souvent nécessaires, que ce soit pour classer ou intégrer des documents, adapter les entrées-sorties d'un service web, construire des réponses cohérentes à partir de données incomplètes, etc. Cet algorithme représente le moyen de tester et de mettre au point un fonctionnement sûr dans de tels contextes. Etant donnés un schéma D, un seuil th et un arbre XML t, il produit *toutes* les séquences d'opérations d'édition-nœud possibles qui transforment t en un document valide et dont le coût ne dépasse pas th. J'en donne d'abord le principe général sur un exemple, avant de rappeler ses propriétés.

1.2.1 Principe

Soit D un schéma représenté par un automate de schéma-élément $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$. Pour présenter le principe je reste dans le cas le plus simple, où D définit un langage d'arbre LTL [11], donc il y a une et une seule règle de transition possible pour chaque nom d'élément dans Σ, donc je peux considérer que $Q = \Sigma$. La figure 1.4 présente les règles de transition de l'automate $\mathcal{A}$ pour lequel $Q = \Sigma = \{root, a, b, c, d\}$ et $Q_f = \{root\}$, en donnant pour chaque expression régulière de ses règles de transition l'automate à états finis correspondant [12]. L'arbre t de la figure 1.1 n'est pas valide par rapport à ce schéma, parce que la séquence abb des étiquettes des nœuds fils du nœud $root$ n'est pas dans le langage $L(b^*|ab^*c)$.

L'objectif est de produire l'ensemble d'arbres valides $\{t'_1, \cdots, t'_n\}$, dont la distance par rapport à t est inférieure ou égale à un seuil th, par exemple $th = 2$. En fait, nous produisons également pour chaque t'_i une séquence d'opérations d'édition-nœud qui permet de dériver t jusqu'à t'_i. Pour cela, nous construisons une *matrice d'édition* M, sur le principe proposé par Selkow [Sel77] pour le calcul de la distance entre deux arbres t et t'. Ce principe consiste lui-même à généraliser la solution de [WF74] pour calculer la distance entre deux mots au cas de deux arbres étiquetés d'arité variable. Trois opérations d'édition sont considérées : (i) changement d'étiquette d'un nœud, (ii) suppression d'un sous-arbre, (iii) insertion d'un sous-arbre. Les deux dernières peuvent se décomposer en séquences de suppressions et (respectivement) d'insertions de nœuds, donc cela se ramène aux trois opérations d'édition-nœud *relabel*, *add* et *delete* de la définition 1.1.10. Un coût est affecté à chacune de ces opérations et le problème résolu par l'algorithme de Selkow est de trouver la séquence d'opérations de plus faible coût qui permet de dériver t' à partir de t et donc la distance entre t et t' (qui est le coût de cette séquence).

Le calcul de cette séquence s'appuie sur une matrice H dont chaque cellule $H[i, j]$ contient la distance d'édition entre deux arbres partiels $t\langle i\rangle$ et $t'\langle j\rangle$ comme l'illustre la figure 1.5(a). Soit $C_{i,j}$ le coût minimal de transformation de $t\langle i\rangle$ en $t'\langle j\rangle$, Selkow a montré

11. L'extension de l'algorithme au cas STTL est décrite dans l'article [ABS13].

12. Selon la définition classique : un automate à états finis est un quintuplet $A = (\Theta, V, \Delta, q_0, F)$ où Θ est l'ensemble fini des états q_i, V est l'alphabet, $q_0 \in \Theta$ est l'état initial, $F \subseteq \Theta$ est l'ensemble des états finaux et $\Delta : \Theta \times V \to \Theta$ est la fonction de transition.

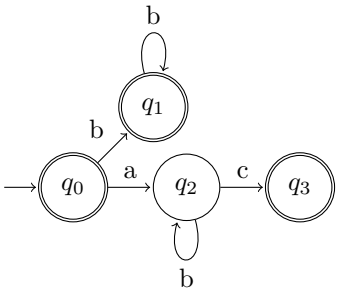
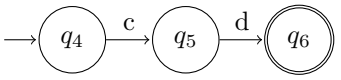
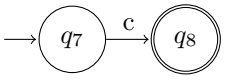
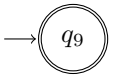
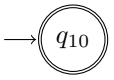
Etiquette	Expression régulière	Automates à états finis	état
root	$b^*\|ab^*c$		root
a	cd		a
b	c		b
c	ϵ		c
d	ϵ		d

Figure 1.4 – Règles de transition de l'automate schéma-élément $\mathcal{A}$.

que $C_{i,j}$ est le coût d'une séquence d'opérations sélectionnée car ayant le coût minimal, parmi les trois suivantes : *(1)* transformer $t\langle i\rangle$ en $t'\langle j-1\rangle$ et insérer le sous-arbre $t'_{|_j}$, *(2)* transformer $t\langle i-1\rangle$ en $t'\langle j-1\rangle$ et transformer le sous-arbre $t_{|_i}$ en $t'_{|_j}$ et *(3)* transformer $t\langle i-1\rangle$ en $t'\langle j\rangle$ et supprimer le sous-arbre $t_{|_i}$. Ceci est illustré en figure 1.5(b) : la matrice H est calculée colonne par colonne, de gauche à droite et de haut en bas. Ainsi chaque cellule $H[i,j]$ est déduite de ses trois voisines $H[i-1,j-1]$, $H[i-1,j]$ et $H[i,j-1]$. Elle contient donc la valeur minimale entre *(1)* la valeur contenue dans la cellule gauche plus le coût minimal d'insertion de $t'_{|_j}$ (Figure 1.5(b), flèche (1)), *(2)* la valeur contenue dans la cellule diagonale gauche plus le coût minimal de transformation de l'arbre $t_{|_i}$ en $t'_{|_j}$ (Figure 1.5 (b), flèche (2)), et *(3)* la valeur contenue dans la cellule au-dessus plus le coût minimal de suppression du sous-arbre $t_{|_i}$ (Figure 1.5 (b), flèche (3)).

Calculer la distance d'édition entre t et t' suppose donc de calculer les distances entre leurs sous-arbres respectifs. Ce calcul est en $O(\Sigma_{i=0}^{min(h_t,h_{t'})} n_i n'_i)$, où d'une part h_t et $h_{t'}$ sont les hauteurs de t et t', d'autre part n_i et n'_i sont les nombres de nœuds à la hauteur i dans t et (respectivement) t'.

Nous utilisons ce calcul de distance entre deux arbres pour déterminer la distance d'un arbre à un *langage d'arbres*. Précisément, pour trouver l'ensemble des séquences d'opérations permettant de dériver des arbres $\{t'_1, \cdots, t'_n\}$ valides par rapport à un schéma D en partant d'un arbre t, tous restant à une distance de t inférieure ou égale à un seuil th, nous calculons *dynamiquement* la matrice M, selon le même processus que celui décrit par

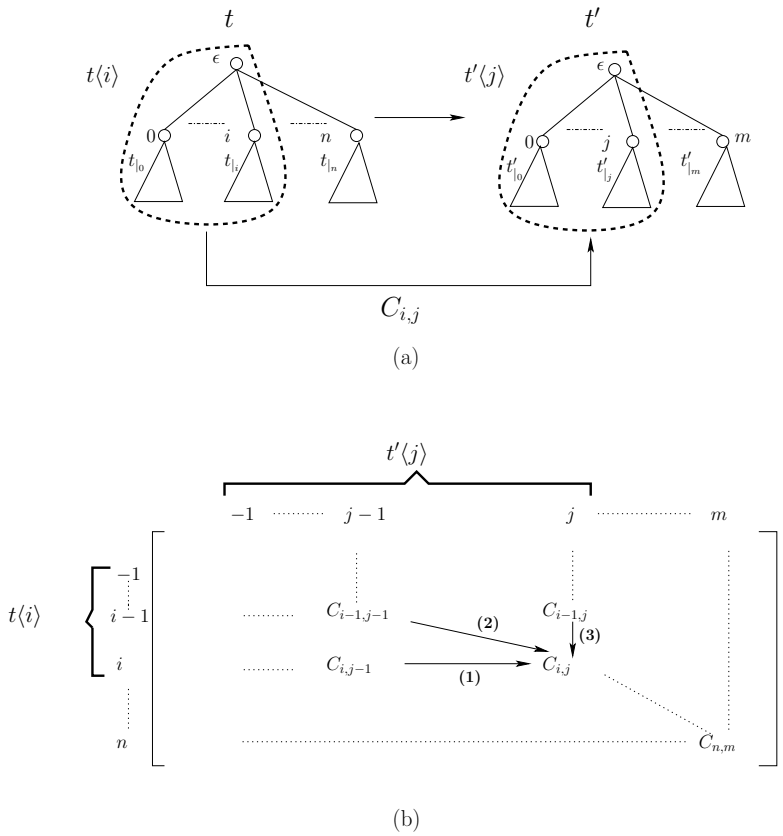


FIGURE 1.5 – (a) Deux arbres partiels $t\langle i\rangle$ et $t'\langle j\rangle$. (b) Matrice de distance entre deux arbres [Sel77] : calcul de la cellule $H[i,j] = C_{i,j}$.

[Ofl96] pour trouver les corrections d'un mot par rapport à un langage de mots. En effet l'algorithme proposé par Oflazer prend en entrée un mot X qui n'appartient pas à un langage de mots représenté par un automate à états finis A et recherche toutes les corrections possibles de ce mot (c'est-à-dire tous les mots reconnus par A) qui sont à une distance de X inférieure ou égale à un seuil donné th. Cela consiste en une exploration dynamique de l'automate à états finis A.

Un mot correct $Y = a_1 a_2 \dots a_k$ est progressivement généré en concaténant les étiquettes des transitions de A à partir de l'état initial q_0 et jusqu'à un état final. A chaque étape, si l'automate est dans son état q_m, pour compléter le mot courant $Y = a_1 a_2 \dots a_k$ par l'étiquette b d'une transition sortante de q_m, l'algorithme teste si la distance d'édition "cut-off" entre X et le mot $Y = a_1 a_2 \dots a_k b$ reste inférieure à th. La distance d'édition "cut-off" entre deux mots est une mesure introduite par [DC92] qui permet d'optimiser le processus d'exploration de l'automate en le stoppant dès qu'il ne peut plus y avoir de nouvel Y dans le seuil th. A ce moment le fil d'exploration courant termine, le caractère b est retiré de Y, il y a retour en arrière vers l'état q_m et l'exploration continue en testant une nouvelle transition sortante de q_m. Lorsqu'un état final est atteint, si $dist(X,Y) \leq th$ alors Y est un candidat valide pour corriger X.

Tout en réalisant cette exploration dynamique de l'automate A, de la même manière que

dans [DC92], Oflazer propose de construire progressivement H, une matrice de distances d'édition entre X et *chacun des candidats potentiels* Y : $H[i, j]$ contient la distance entre les préfixes de longueur i et j des deux mots X et (respectivement) Y. La plus-value de la proposition de Oflazer dans [Ofl96] réside dans la représentation du lexique par l'automate A de sorte que, lorsqu'un mot est recherché dans le lexique, les colonnes de la matrice qui correspondent à un même préfixe de mots du lexique ne sont calculées qu'une seule fois.

Pour résumer, l'algorithme de correction de documents que nous avons mis au point combine les propositions fondamentales de [Sel77] et de [Ofl96]. Nous utilisons le calcul de distance d'édition entre deux arbres de Selkow, défini pour les trois opérations d'édition-nœud *relabel*, *add* et *delete*, pour calculer dynamiquement les distances entre l'arbre partiel courant de t et un arbre partiel généré progressivement par une exploration dynamique des automates à états finis présents dans les règles de transition de l'automate de schéma $\mathcal{A}$. L'arbre partiel joue ici pour les arbres le rôle que joue le préfixe pour les mots.

Revenons à notre exemple : la matrice M contient l'ensemble des séquences d'opérations, dont le coût ne doit pas dépasser th, qui transforment les arbres partiels de t en arbres partiels de t'_i. Il peut y avoir plusieurs corrections. $M[i][j]$ ou (i, j) dénote la cellule de la matrice en ligne i et colonne j. La cellule $(0, 0)$ de la matrice contient la séquence d'opérations qui transforme la racine de t en la racine attendue pour un arbre valide, c'est-à-dire appartenant à $L(\mathcal{A})$, le langage des arbres reconnus par $\mathcal{A}$. Dans l'exemple, c'est-à-dire l'arbre t de la figure 1.1 et l'automate $\mathcal{A}$ dont les règles de transition sont en figure1.4, t a à sa racine l'étiquette attendue comme racine par $\mathcal{A}$, donc cette cellule reçoit la séquence d'opérations vide $\{nos_\emptyset\}$, comme illustré en figure 1.6. La matrice M a autant de lignes que l'arbre à corriger t a de fils directs de sa racine : dans notre exemple cela fait donc 3 lignes.

	M	0 root	1 b	2 b	3 b	**4** ***b***
0	root	$\{nos_\emptyset\}$	$\{\langle(add, 0, b), (add, 0.0, c)\rangle\}$	$\emptyset$	$\emptyset$	$\boldsymbol{\emptyset}$
1	a	$\emptyset$	$\{os_1{=}\langle(relabel, 0, b), (delete, 0.1, /)\rangle\}$	$\emptyset$	$\emptyset$	$\boldsymbol{\emptyset}$
2	b	$\emptyset$	$\emptyset$	$\{os_1\}$	$\emptyset$	$\boldsymbol{\emptyset}$
3	b	$\emptyset$	$\emptyset$	$\emptyset$	$\{os_1\}$	$\boldsymbol{\emptyset}$

Figure 1.6 – Contenu de la matrice M

Comme pour le calcul de distance entre deux arbres proposé par Selkow, les autres cellules de M sont calculées progressivement colonne par colonne et de haut en bas. Pour chaque cellule, le calcul consiste à (i) concaténer chaque séquence d'opérations d'édition de la cellule voisine (à gauche, en diagonale-gauche, ou au-dessus) avec l'une des trois opération d'édition-arbre suivantes, à condition que le coût du résultat ne dépasse pas le seuil th :

(i) Ajout de sous-arbre ($\rightarrow$) à partir du contenu de la cellule à gauche : ses séquences d'opérations sont concaténées avec celle correspondant à l'insertion du sous-arbre t'_i (selon la définition 1.1.12). En général différents sous-arbres localement valides peuvent être ajoutés, du moment qu'ils ont à leur racine l'étiquette attendue par l'automate à état fini courant. Pour cette raison, il y a dans les cellules des ensembles

de séquences d'opérations d'édition-nœud.

(ii) Correction de sous-arbre ($\searrow$) à partir du contenu de la cellule au-dessus à gauche : ses séquences d'opérations sont concaténées avec celles correspondant à la correction (cf. définition 1.1.12) d'un sous-arbre de t en un sous-arbre localement valide de l'arbre candidat en cours de construction. Cela se fait donc par un appel récursif à la fonction de correction, qui construit une autre matrice de distance d'édition.

(iii) Suppression d'un sous-arbre ($\downarrow$) : à partir du contenu de la cellule au-dessus : ses séquences d'opérations sont concaténées avec celles correspondant à la suppression d'un sous-arbre de t (selon la définition 1.1.12).

Ainsi dans la figure 1.6, depuis la cellule $(0,0)$ est calculée la cellule $(1,0)$ qui contient la séquence d'opérations pour supprimer le sous-arbre $t_{|0}$, laquelle séquence a un coût égal à 3. De ce fait le seuil est dépassé et cette cellule reçoit $\emptyset$, ainsi que toutes celles qui sont en dessous. Cette première colonne représente en fait la correction de la racine de l'arbre t à corriger.

Chaque nouvelle colonne est ajoutée en suivant une transition dans l'automate à états finis correspondant à la racine (figure 1.4) : par exemple pour la colonne $j = 1$ on peut suivre la transition (q_0, b, q_1) (cette colonne est marquée par l'étiquette b en figure 1.6). Les sous-arbres en position 0 dans l'arbre candidat auront l'étiquette b à leur racine. Les cellules de chaque nouvelle colonne sont calculées de haut en bas, par exemple pour la cellule $(1,1)$ de la figure 1.6) :

(i) La cellule à gauche, $M[1][0]$, contient $\emptyset$ donc aucune séquence d'opération.

(ii) La cellule en haut à gauche, $M[0][0]$, contient $nos_\emptyset$, c'est-à-dire une séquence dont le coût 0. Lui est concaténée la séquence d'opérations $os_1 = \langle (relabel, 0, b), (delete, 0.1, /) \rangle$ qui résulte de la correction du sous-arbre $\{(\epsilon, a), (0, c), (1, d)\}$ en position 0 dans t en un sous-arbre valide ayant b à sa racine. Le sous-arbre valide obtenu est $\{(\epsilon, b), (0, c)\}$. Le coût de os_1 est $2 \leq th = 2$ donc os_1 est conservée dans la cellule $(1,1)$. La matrice calculée pour corriger le sous-arbre $\{(\epsilon, a), (0, c), (1, d)\}$ en $\{(\epsilon, b), (0, c)\}$ est montrée en figure 1.7. La séquence os_1 provient de la séquence présente dans la cellule $(2,1)$ de cette matrice, dans laquelle les positions sont *préfixéee* avec la position 0 (d'où a été lancé l'appel récursif à la fonction de correction).

M'	0 b	1 c
0 a	$\{\langle (relabel, \epsilon, b) \rangle\}$	$\{\langle (relabel, \epsilon, b),$ $(insert, 0, c) \rangle\}$
1 c	$\{\langle (relabel, \epsilon, b),$ $(delete, 0, /) \rangle\}$	$\{\langle (relabel, \epsilon, b) \rangle\}$
2 d	$\emptyset$	$\{\langle (relabel, \epsilon, b),$ $(delete, 1, /) \rangle\}$

FIGURE 1.7 – Autre matrice calculée par l'appel récursif à la fonction de correction.

(iii) La cellule au-dessus, $M[0][1]$, contient la séquence $\langle (add, 0, b), (add, 0.0, c) \rangle$ dont le coût est égal à 2. Lui est concaténée la séquence $os_2 = \{\langle (delete, 0.1, /), (delete, 0.0, /),$ $(delete, 0, /) \rangle\}$, de suppression du sous-arbre en position 0 dans t. Cependant le coût de cette suppression est 3 donc la concaténation résulte en une séquence de coût 5,

qui excède le seuil égal à 2.

Les étiquettes de la séquence des colonnes $(0 < j)$ dans M forment un mot u, par exemple $u = bbbb$ pour la matrice de la figure 1.6. Ce mot est forcément un préfixe d'un mot valide puisqu'il est construit en parcourant l'automate. Si le mot u appartient à $L(A_a)$ (c'est-à-dire si l'état courant est un état final de l'automate courant) alors la dernière cellule en bas de la colonne contient les séquences d'opérations qui permettent de dériver un arbre candidat t' à partir de t. Dans notre exemple, aussi bien l'état initial que l'état atteint en suivant la transition étiquettée b sont des états finaux. Il n'y a rien dans les cellules du bas des colonnes 1 et 2 de la matrice de la figure 1.6, par contre la cellule $(3, 3)$ contient la séquence os_1=$\langle(relabel, 0, b), (delete, 0.1, /)\rangle$, qui peut donc transformer t en un arbre valide, lequel est l'arbre t'_1 de la figure 1.8.

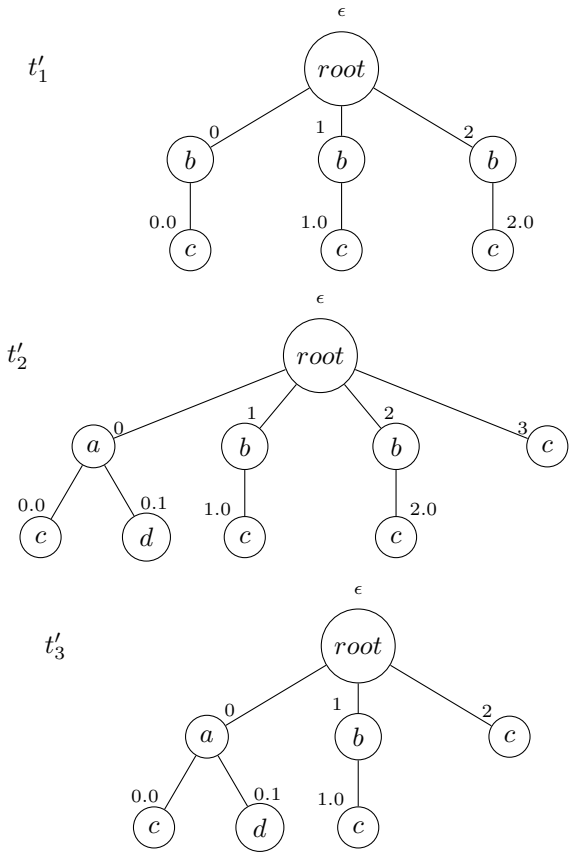


Figure 1.8 – Trois corrections possibles pour l'arbre t de la figure 1.1

.

Par ailleurs, lorsque toutes les cellules d'une colonne sont vides alors il est inutile de continuer le fil d'exploration de l'automate courant : il y a retour en arrière pour essayer systématiquement les autres transitions possibles. C'est le cas par exemple pour la colonne 4 de la matrice de la figure 1.6. ici le retour arrière va supprimer les colonnes 4, 3, 2 et 1, ce qui correspond à un retour à l'état q_0 de l'automate associé à l'étiquette *root*, à partir

duquel il est possible de suivre la transition (q_0, a, q_2). Les 2 autres corrections possibles montrées en figure 1.8 sont obtenues dans la suite de cette exploration systématique, coupée régulièrement grâce au seuil th qui amène les cellules à devenir vides dès que plus aucune séquence n'a un coût inférieur.

Nous avons publié l'algorithme complet de *recherche exhaustive de toutes les séquences d'édition de coût inférieur à un seuil donné permettant de transformer un arbre XML en un arbre valide par rapport à une DTD donnée* dans [ABS13]. Nous y présentons (i) l'algorithme en lui-même, (ii) l'ensemble des définitions nécessaires à sa formulation et à son analyse, (iii) la preuve de ses propriétés, (iv) l'analyse des résultats des expérimentations réalisées sur des données réelles à grande échelle à partir d'une implémentation complète [13], (v) ses extensions directes, notamment au cas de schémas reconnaissant les langages d'arbre à type unique (STTL) et enfin (vi) sa situation par rapport à un état de l'art approfondi concernant le calcul de distances et de transformations de documents par rapport à des schémas. Je synthétise tout cela rapidement dans la section suivante.

1.2.2 Propriétés

L'algorithme pour corriger un arbre t par rapport à un schéma D, représenté par un automate de schéma-élément $\mathcal{A}$, en respectant un seuil th, utilise une méthode de programmation dynamique qui calcule progressivement la matrice de distance d'édition entre l'arbre t et le langage d'arbres reconnu par $\mathcal{A}$, notée M_u^c, où c est l'étiquette attendue à la racine de l'arbre corrigé et $u = u_1 u_2 \dots u_k$ est un mot d'étiquettes. Chaque cellule $M_u^c[i][j]$ contient l'ensemble de toutes les séquences d'édition-nœud qui permettent de dériver les arbres $t_1, \dots, t_n$ à partir de $t\langle i{-}1\rangle$, chaque t_j $(1{\leq}j{\leq}n)$

- étant partiellement valide ;
- ayant l'étiquette c à sa racine et sous cette racine un mot d'étiquettes des fils formant un préfixe [14] de u de longueur j ;
- étant à une distance de t inférieure ou égale à th.

Avec les notations introduites dans les définitions précédentes cela s'écrit :
$\exists_{t_1,\dots,t_n}[t\langle i{-}1\rangle \overset{M_u^c[i][j]}{\longrightarrow} \{t_1, \dots, t_n\}$ and $\forall_{1\leq k\leq n}[t_k \in L^c_{u[1\dots j]}(S)$ et $dist(t, t_k) \leq th]]$.
Ainsi pour c égal à une étiquette de racine valide, à la ligne $i = \bar{t}$ et à la colonne $j = |u|$ la cellule $M_u^c[i][j]$ contient l'ensemble de toutes les séquences d'édition-nœud qui permettent de dériver à partir de l'arbre t l'ensemble des arbres reconnus par $\mathcal{A}$, distants d'au plus th de t. Nous notons $L_t^{th}(\mathcal{A})$ cet ensemble. Les propriétés suivantes sont démontrées dans l'article :

Lemme 1 *Etant donnés l'étiquette $c \in \Sigma$, le schéma D représenté par l'automate $\mathcal{A}$ et le seuil th, soit $\mathcal{A}'$ l'automate identique à $\mathcal{A}$ sauf que l'ensemble Q_f de $\mathcal{A}'$ contient l'état associé à c, un appel à la fonction correction($t_\emptyset$,$\mathcal{A}$,th,c) termine et retourne l'ensemble Result tel que : $t_\emptyset \overset{Result}{\longrightarrow} L^{th}_{t_\emptyset}(\mathcal{A}')$.*

13. Le prototype développé, sa documentation, sa licence et les données réelles utilisées sont libres d'accès à l'adresse : http ://www.info.univ-tours.fr/%7Esavary/English/xmlcorrector.html.

14. Pour $u = u_1 u_2 \dots u_j \dots u_n \in \Sigma^*$, $|u|$ dénote la longueur de u, i.e. $|u| = n$ et $u[1..j]$ dénote le préfixe de u de longueur j.

Le lemme 1 établit que la fonction *correction* permet de "corriger" un arbre vide en proposant un ensemble de dérivations possibles qui mènent à un ensemble d'arbres localement valides. Corriger l'arbre vide vers un arbre ayant pour racine c représente l'opération d'insertion d'un sous-arbre ayant c pour racine.

Lemme 2 *Etant donnés l'étiquette $c \in \Sigma$, le schéma D représenté par l'automate $\mathcal{A}$ et l'arbre t, soit $\mathcal{A}'$ l'automate identique à $\mathcal{A}$ sauf que l'ensemble Q_f de $\mathcal{A}'$ contient l'état associé à c, un appel à la fonction correction(t,$\mathcal{A}$,0,c) termine et retourne l'ensemble Result tel que : $t \overset{Result}{\longrightarrow} L_t^0(\mathcal{A}')$.*

Clairement, $L_t^0(\mathcal{A}') = \{t' \mid t' \in L(\mathcal{A}') \text{ et } dist(t,t') = 0\}$, donc cet ensemble contient uniquement t si t est localement valide et rien sinon. Le lemme 2 indique donc simplement que la fonction *correction* permet de décider si un sous-arbre est localement valide ou pas.

Lemme 3 *Etant donnés l'étiquette $c \in \Sigma$, le schéma D représenté par l'automate $\mathcal{A}$, l'arbre $t \neq t_\emptyset$ et le seuil $th > 0$, soit $u \in \Sigma^*$ un mot tel que $u \in L(FSA_{\mathcal{A}.Q_f})$ et $L_u^c(\mathcal{A}) \neq \emptyset$, un appel à la fonction correction$(t, \mathcal{A}, th, c)$ calcule la matrice M_u^c telle que, pour chaque $0 \leq i \leq \bar{t}$ et pour chaque $0 \leq j \leq |u|$, on a : $t\langle i-1\rangle \overset{M_u^c[i][j]}{\longrightarrow} \{t' \mid t' \in L^c_{u[1..j]}(\mathcal{A}), dist(t\langle i-1\rangle, t') \leq th\}$.*

Le lemme 3 établit que chaque cellule de la matrice M_u^c contient toujours un ensemble de séquences d'opérations d'édition de coût inférieur ou égal à th, chacune permettant de dériver un arbre partiellement valide t' dont les fils de la racine ont des étiquettes formant un préfixe de u. A partir de ces lemmes le théorème suivant est démontré :

Théorème 1.2.1 Etant donnés le schéma D représenté par l'automate $\mathcal{A}$ pour lequel l'étiquette de racine attendue est $root$, l'arbre t et le seuil $0 \leq th$, un appel à la fonction $correction(t, \mathcal{A}, th, root)$ termine et retourne l'ensemble $Result$ tel que : $t \overset{Result}{\longrightarrow} L_t^{th}(\mathcal{A})$.

Il est également démontré dans l'article que la complexité en temps de l'algorithme de correction est en $O((f_t + 1) \times (f_\mathcal{A})^{|t|+th} \times 6 \times |\Sigma| \times (|t| + th)^{th})$, où f_t est le nombre maximum de fils d'un nœud de t, $f_\mathcal{A}$ est le maximum de transitions sortantes parmi tous les automates à états finis des transitions de $\mathcal{A}$, $|t|$ est le nombre de nœuds de t, Σ est l'alphabet de $\mathcal{A}$ et th est le seuil de correction. Cette complexité exponentielle vient du fait que l'ensemble des séquences d'opérations d'édition et les arbres corrects correspondants sont effectivement tous générés.

En pratique deux faits limitent cependant cette complexité, qui sont difficiles à formaliser dans l'analyse de complexité : d'une part l'exploration dynamique des automates à états finis est rapidement stoppée par le seuil th car on arrête d'ajouter une colonne dès que toutes les cellules de la colonne courante sont vides (une cellule est vide dès que toutes ses séquences d'opération d'édition potentielles ont un coût supérieur à th) ; d'autre part, comme dans la proposition de Oflazer pour les mots, les calculs des arbres partiels sont factorisés : les premières colonnes sont mutualisées pour les mêmes préfixes. Cela peut expliquer les comportements clairement plus polynomiaux qu'exponentiels observés lors des expérimentations poussées que nous avons menées.

Ces expérimentations ont été réalisées avec des données réelles et divers scénarios ont été construits pour tester les performances en fonction des critères suivants : (i) la taille du document, (ii) la valeur du seuil de correction, (iii) le nombre d'erreurs, (iv) la position d'une erreur dans le document et (v) la forme du schéma. L'article présente les scénarios et leurs résultats accompagnés d'une analyse détaillée et, surtout, ces résultats peuvent être reproduits en utilisant le prototype (sous licence GNU LGPL v3) et les données que nous avons utilisées (cf. note de bas de page numéro 13).

L'article présente également un état de l'art approfondi assorti d'une étude contrastive des propositions plus ou moins comparables à la nôtre, allant du simple calcul de distance entre un document et un schéma ([BGM04, BGM08], [TCY07], [XMXV06], [SC06, SFC08], [TVY08]) au véritable calcul d'un ensemble de corrections (séquences d'opérations d'édition) du document par rapport au schéma (l'ensemble étant plus ou moins réduit selon les cas [Svo10, SM11, Suz07]), en passant par le calcul d'une unique correction, annoncée comme minimale alors que sa découverte repose sur des heuristiques ([BdR04]).

Notre approche est unique dans son aspect *complet* : calcul de *tous* les arbres valides t' à distance inférieure ou égale à th d'un arbre donné t. La garantie d'avoir toutes les solutions satisfaisant un certain critère (ici la distance à l'arbre t) a l'avantage d'assurer que la "meilleure" solution (satisfaisant ce critère) est dans l'ensemble de solutions trouvées. Cette notion de "meilleure" solution dépend entièrement du contexte de l'application et est souvent informelle donc difficile à sélectionner automatiquement a priori. Notre proposition n'est pas de présenter un grand ensemble de solutions à un utilisateur final, ce qui est évidemment contre-productif, mais de permettre aux développeurs d'applications de réaliser les tests nécessaires, selon des scenarios précis, de façon à adapter le calcul à leur contexte particulier, que ce soit avec des pré-traitements, des post-traitements ou en modifiant l'algorithme de correction en lui-même (les sources sont librement utilisables et modifiables).

Par exemple, lorsqu'une mise à jour de schéma invalide un ensemble de documents, la plupart d'entre eux vont nécessiter la même correction : l'administrateur peut alors utiliser notre code pour rechercher la solution la plus pertinente pour un document représentant l'ensemble, en fonction du contexte particulier de l'application. La séquence d'opérations d'édition choisie peut ensuite être généralisée à tout l'ensemble de documents.

Puisque l'algorithme trouve toutes les corrections dans un seuil de distance donné, la question du choix de la valeur du seuil est incontournable et dépend, elle aussi, de l'application particulière dans laquelle la correction est recherchée : si le seuil est inutilement grand le temps de calcul sera important et le nombre de corrections probablement aussi ; mais si le seuil est trop faible des corrections intéressantes peuvent être ignorées... L'algorithme lui-même présente l'intérêt de pouvoir être utilisé pour trouver une valeur de seuil utile car pour des seuils faibles les calculs s'arrêtent vite : on peut donc rechercher le seuil pas-à-pas. Si le document n'est pas valide alors il est à une distance au moins égale à 1 du schéma, on peut donc rechercher des corrections avec $th = 1$; s'il n'y en a pas alors on peut recommencer avec $th = 2$, etc [15]. Encore une fois nous offrons la garantie d'avoir à chaque étape toutes les corrections dans le seuil.

15. Notre prototype applique ce scénario lorsque l'utilisateur demande seulement une correction de coût minimal.

Les avantages et inconvénients de calculer toutes les solutions à un problème, satisfaisant un critère précis, sont l'objet d'études dans beaucoup d'autres domaines. C'est le cas par exemple pour le problème de la détermination de correspondances approximatives entre mots comme le montre le tour d'horizon présenté dans [Boy11] où il est montré à quel point ce problème a reçu de solutions provenant de communautés scientifiques différentes, qui ont souvent construit des algorithmes comparables en parallèle, chacune pour leur contexte spécifique et parfois avec l'objectif de complétude atteint dans notre proposition. Etant donné l'importance prise désormais par XML dans la plupart des systèmes d'information, cette complétude est un avantage réellement important pour mettre au point des systèmes d'interrogation, d'intégration et d'évolution maîtrisés.

La question de la spécification et de la validation des contraintes d'intégrité rejoint ce souci de maîtrise par des solutions de contrôle automatique, cette fois non plus de la structure mais *des données* décrites et échangées via des documents XML. C'est l'objet d'une autre de nos contributions, exposée dans la section suivante.

1.3 Des contraintes d'intégrité

Les contraintes d'intégrité représentent des propriétés qui doivent être satisfaites par toutes les instances d'une base de données. Elles peuvent s'exprimer en logique du premier ordre, mais pour que les systèmes de gestion de bases de données puissent les vérifier automatiquement de façon efficace, les recherches se sont orientées vers l'étude de classes plus restreintes, appelées *dépendances* [AV85, AHV95]. Par exemple, dire que le numéro de sécurité sociale (NSS) est la clé d'une relation EMPLOYEE est une contrainte d'intégrité qui impose l'existence et l'unicité des valeurs NSS dans toute la relation EMPLOYEE. Les contraintes de clés sont un cas particulier de contraintes plus générales appelées dépendances fonctionnelles. Un exemple d'une dépendance fonctionnelle qui n'est pas forcément une clé est la contrainte imposant qu'une même adresse ne puisse avoir qu'un seul code postal (exprimée par l'expression $Address \rightarrow ZIPCode$). Les dépendances d'inclusion représentent un autre type de contrainte d'intégrité important. Par exemple, en supposant que la relation EMPLOYEE comprenne le département *Dept* de chaque employé et qu'une table DEPARTEMENT stocke des informations pour chaque département de l'entreprise, une contrainte d'inclusion peut imposer que toute valeur *Dept* stockée dans EMPLOYEE doive exister dans DEPARTEMENT.

De cette manière le système d'information reste cohérent et en ceci les contraintes d'intégrité sont un aspect essentiel de la *qualité des données*. Les contraintes d'intégrité jouent un rôle tout aussi essentiel au niveau de la qualité des *schémas* de données, en servant de base à la théorie des formes normales. La possibilité de raisonnement formel sur les contraintes d'intégrité [BB79, Mai80, AHV95, LL99] est également un facteur de qualité pour les bases de données. Un autre avantage qu'il y a à pouvoir définir des contraintes d'intégrité est qu'elles représentent des informations qui mènent à des implantations de stockage, de requêtes et de mises à jour plus efficaces.

Dans le cadre XML, les principes de conception de documents et de schémas, de cohérence des données XML et de maintien de cette cohérence sont également importants et font encore l'objet de nombreux travaux. L'étude des contraintes d'intégrité pour XML a

commencé au début des années 2000, et la plupart des contraintes d'intégrité ont désormais été abordées, analysées (implication et axiomatisation) et appliquées [BDF+01, BFSW01, FSW01, LLL02, BDF+03, HS03, LVL03, AL04, VLL04, DT05, Fan05, Sch05, WT05, HT06, Hal07, VLM07, YJ08, HKL+08, Are09, GI10, LY08, ZXZ09, HL10, HLT10, TZ11, VLM12]. Ces résultats ont démontré à la fois l'intérêt qu'il y a à exprimer des contraintes d'intégrité sur les données XML (spécification), l'intérêt de les définir formellement (analyse, inférences) et l'intérêt de les vérifier automatiquement.

1.3.1 Cadre générique de spécification

Concernant la *spécification* de contraintes d'intégrité pour XML, un exemple est le formalisme XIC [DT05] dans lequel une contrainte d'intégrité est définie ainsi : $\forall x_1, \ldots, x_n$ $A(x_1, \ldots, x_n) \rightarrow \exists y_1, \ldots, y_m$ $B(x_1, \ldots, x_n, y_1, \ldots, y_m)$, où x_i et y_j dénotent des variables dont les valeurs sont les noms des éléments ou attributs du document, et A et B sont des conjonctions d'atomes $x = y$ ou x *path* y, *path* définissant une relation de chemins entre x et y. Comme l'indiquent les auteurs, comme dans le cadre relationnel, le pouvoir d'expression entraîne des problèmes d'indécidabilité pour des problèmes pourtant extrêmement utiles, comme l'inclusion. A l'opposé, les formalismes DTD, XML-Schema ou Relax-NG, conçus pour la définition de contraintes de structure, offrent peu de constructions pour la spécification de contraintes d'intégrité. Les limites des attributs $ID/IDREF(S)$ d'une DTD sont évidentes : un attribut ID doit avoir une valeur unique, mais parmi tous les attributs ID du document XML, ce qui empêche par exemple des composants et des fournisseurs d'avoir des identifiants identiques. De même, une référence $IDREF$ peut correspondre à n'importe quel ID dans le document. De son côté, Relax-NG ne traite que de la structure. Le langage XML-Schema permet d'exprimer au moins des clés et clés étrangères.

Les contraintes de schéma et les contrainte d'intégrité peuvent être définies et vérifiées indépendamment les unes des autres. Dans [AL04], la définition de contrainte d'intégrité est fondée sur une DTD, ce qui leur permet de définir assez naturellement une notion de *forme normale* de documents XML. Pour autant, les mêmes auteurs ont démontré que la vérification automatique de la consistance entre contraintes de schéma et contraintes d'intégrité est en général indécidable [AFL02a, AFL02b]. La majorité des propositions concernant les contraintes d'intégrité ne présupposent pas l'existence d'un schéma. Par contre toutes spécifient les contraintes à l'aide *d'expressions de chemins*, expressions régulières simples sur l'alphabet des noms d'éléments, ces expressions de chemins représentant d'une manière ou d'une autre également des formes de contraintes structurelles, plus souples que les schémas définis pour XML. C'est également le cas dans notre proposition.

Dans [BAL09], nous avons proposé un cadre homogène pour exprimer différentes sortes de contraintes d'intégrité, que nous avons repris dans [BFL12] en l'étendant à de nouvelles contraintes proposées dans la littérature. Ce cadre est générique, en particulier il prend en compte la notion de spécification *absolue* ou *relative* (contextuelle) ainsi que la distinction entre les deux sortes d'égalité possibles entre noeuds : l'égalité *de valeur* et l'égalité de noeud. Ces aspects génériques étaient présents dans le papier précurseur [BDF+01] mais ne se retrouvent pas toujours dans les propositions qui ont suivi [LVL03, AL04, WT05]. Je rappelle brièvement notre formalisme générique et comment chaque sorte de contrainte est concrètement définie.

Tout d'abord, les contraintes d'intégrité pour XML sont définies sur les documents, c'est-à-dire les arbres XML, plus précisément sur les *valeurs* situées aux feuilles des arbres. La définition suivante établit comment les arbres et leurs valeurs sont utilisés dans ce cadre.

Définition 1.3.1 Arbre XML étendu pour les contraintes d'intégrité Soit $\Sigma = \Sigma_{ele} \cup \Sigma_{att} \cup \{data\}$ un alphabet où Σ_{ele} est l'ensemble des noms d'éléments, Σ_{att} est l'ensemble des noms d'attributs et *data* représente les données. Un arbre XML étendu est un triplet $\mathcal{T} = (t, type, value)$, où l'arbre t respecte la définition 1.1.1 [16] et, étant donnée une position p, la fonction $type(t, p)$ retourne l'une des trois valeurs $\{data, element, attribute\}$. De même, $value(t, p) = \begin{cases} p & \text{si } type(t, p) = element \\ val \in \mathbf{V} & \text{sinon, } \mathbf{V} \text{ étant un domaine récursivement énumérable.} \end{cases}$

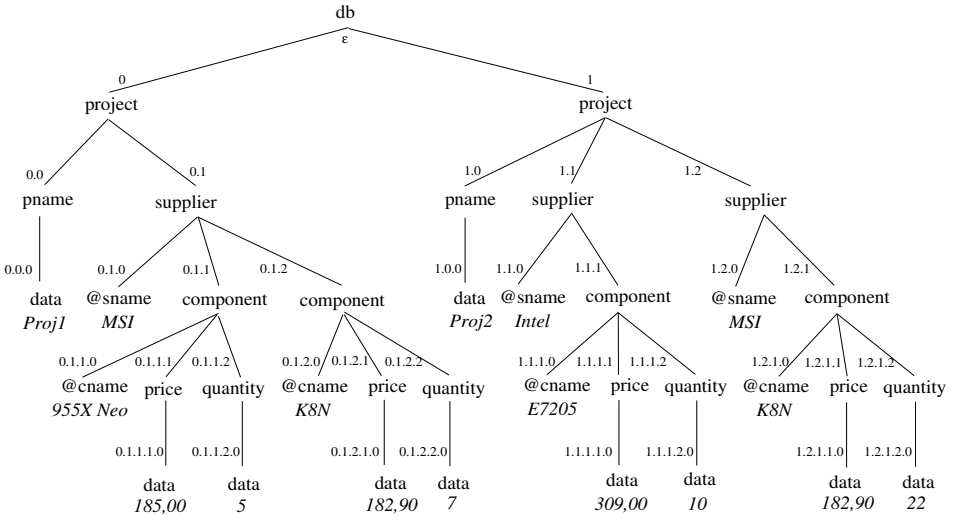


Figure 1.9 – Arbre XML contenant des valeurs concernant les composants impliqués dans des projets.

La figure 1.9 représente un arbre XML étendu, avec pour chaque nœud sa position, son étiquette et les valeurs (en italique) associées aux feuilles. Les attributs y sont représentés précédés du symbole @.

Un chemin dans un arbre XML est une séquence d'éléments de $\Sigma = \Sigma_{ele} \cup \Sigma_{att} \cup \{data\}$. Nous utilisons le langage très simple PL, proposé dans [BDF+01, BDF+03] pour la définition de clé, dont la syntaxe est rappelée dans la table suivante.

Langage de chemin	**Syntaxe**
PL_s	$\rho ::= \ l \mid \rho/\rho \mid \_$
PL	$\upsilon ::= [\,] \mid \upsilon/[\,] \mid [\,]/\upsilon \mid \rho \mid \upsilon//\rho$

Dans la définition de PL, $[\,]$ représente le chemin vide, l est un élément de Σ (étiquettes des nœuds de l'arbre XML étendu ou *data*), le symbole "/" est l'opération de concatenation,

16. La définition 1.1.1 permet de tenir compte du caractère *ordonné* de l'arbre des *éléments* XML : nous verrons que pour les contraintes d'intégrité l'ordre n'est pas indispensable.

"//" représente une séquence quelconque d'étiquettes (éventuellement vide), et "_" dénote n'importe quel nom dans $\Sigma_{ele} \cup \Sigma_{att}$. Le langage PL_s est un intermédiaire pour définir PL. Nous appellerons *chemin simple* un chemin de PL qui ne contient pas "//". Un chemin de PL décrit un ensemble de chemins simples. Un chemin P est *valide* s'il est conforme à la syntaxe de PL_s ou PL et si pour chaque nom $l \in P$, si $l = data$ ou $l \in \Sigma_{att}$, alors l est le dernier symbole de P. Un chemin P définit un automate à états finis A_P dont l'alphabet d'entrée est Σ, comme le précise la définition suivante.

Définition 1.3.2 Automate à états finis A_P d'un chemin P : étant donné un chemin P valide, définit sur l'alphabet Σ, l'automate à états finis A_P est un quintuplet $(Q, \Sigma, I, \delta, F)$ où Q est un ensemble fini d'états, s_0 est l'état initial, δ est la fonction de transition $\delta : Q \times (\Sigma \cup \{any\}) \rightarrow Q$ et F est l'ensemble d'états finals. A_P est construit à partir de P en commençant avec $Q = \{s_0\}$. Soit s l'état courant dans A_P, si P est le chemin vide alors $F = \{s_0\}$, sinon tant qu'on n'est pas à la fin de P, soit $\mathcal{S} = \Sigma \cup \{\_\} \cup \{[\,]\}$ et soit a le symbole suivant dans P (appartenant à $\mathcal{S}$) :
(1) si $a \in \Sigma$ est un symbole isolé ou s'il est à la droite d'un symbole '/' (c'est-à-dire, $/a$), alors un nouvel état s_i ($i \in \mathbb{N}$) est ajouté à Q et la transition $\delta(s, a) = s_i$ est définie. L'état $s = s_i$ est alors l'état courant dans A_P.
(2) si $a \in \Sigma$ apparaît à droite de '//' (c'est-à-dire, $//a$), un nouvel état s_i est ajouté à Q et la transition $\delta(s, a) = s_i$ est définie. La transition $\delta(s, any) = s$ est également définie, où *any* représente n'importe quel symbole de Σ. L'état $s = s_i$ est alors l'état courant dans A_P.
(3) si $a =$'_', un nouvel état s_i est ajouté à Q et la transition $\delta(s, any) = s_i$ est définie, où *any* représente n'importe quel symbole de Σ. L'état $s = s_i$ est alors l'état courant dans A_P.
(4) si $a =$'[ ]' apparaît à droite de '/', on continue simplement le parcours de P.
A la fin de P, l'état courant est ajouté à F.

Les *instances d'un chemin P dans un arbre XML t* sont les séquences de positions dans t dont les noms forment un mot qui appartient au langage reconnu par A_P. Nous dénotons "/" l'opérateur de séquence aussi bien pour les séquences de noms que pour les séquences de positions, alors que nous utilisons le point (".") pour écrire les positions comme par exemple 1.2.0. Soit s une séquence de positions, la fonction $label(s)$ fournit la séquence des noms associés aux positions de s : si $s = p_1/p_2/\ldots/p_n$ alors $label(s) = t(p_1)/t(p_2)/\ldots/t(p_n)$.

Définition 1.3.3 Instance d'un chemin P sur un arbre t : Soit P un chemin en PL, A_P l'automate à états finis correspondant à P et $L(A_P)$ le langage accepté par A_P. Soit $I = v_1/v_2/\ldots/v_n$ une séquence de positions dans t telle que chaque v_i est un fils direct de v_{i-1}. I est une instance de P sur t si et seulement si $label(I) \in L(A_P)$.

Par exemple, le chemin $bd//supplier$ a pour instances $\epsilon/0/0.1$ et $\epsilon/1/1.1$ sur l'arbre de la figure 1.9. Ces mêmes séquences de positions sont également instances du chemin $bd/project/supplier$.

Soit $\mathbb{P}$ l'ensemble fini de tous les chemins simples qui peuvent être instanciés sur un arbre t. Nous définissons le motif M de $\mathbb{P}$ comme étant un ensemble de chemins dont les

plus longs sont ceux de $\mathbb{P}$, qui tous partagent un préfixe commun et pour tout chemin $P \in M$, si P_1 est un préfixe de P, alors $P_1 \in M$.

Exemple 1.3.1 Soit $\{/db//supplier/@sname$, $/db//supplier/$ $component/@cname$, $/db//supplier/component/price\}$ un ensemble de chemins sur le documents de la figure 1.9. Cet ensemble de chemins définit le motif suivant : $\{/db/projet/supplier/@sname$, $/db/project/supplier/compo\text{-}nent/@cname$, $/db/project/supplier/component/price$, $/db$, $/db/projet$, $/db/projet/supplier/$, $/db/project/supplier/component/\}$. □

Nous appelons *instances de motif* les ensembles d'instances des chemins d'un motif sur t qui sont fermés par préfixe sur les séquences de positions. Par exemple l'ensemble $\{\epsilon/0/0.1/0.1.0$, $\epsilon/0/0.1/0.1.1/0.1.1.0$, $\epsilon/0/0.1/0.1.1/0.1.1.1\}$ est une instance du motif de l'exemple 1.3.1, alors que ce n'est pas le cas pour les ensembles $\{\epsilon/0/0.1/0.1.0$, $\epsilon/0/0.1/0.1.1/0.1.1.0$, $\epsilon/0/0.1/0.1.2/$ $0.1.2.1\}$ et $\{\epsilon/0/0.1/0.1.0$, $\epsilon/1/1.1/1.1.1/1.1.1.0$, $\epsilon/1/1.1/1.1.1/$ $1.1.1.1\}$. La définition suivante précise ces notions très importantes pour fixer la sémantique des contraintes énoncées dans le cadre générique que nous proposons.

Définition 1.3.4 Motif et instance de motif : Un *motif* M est un ensemble fini de chemins fermés par préfixe [17] dont les instances existent dans un arbre t. Soit $Long_M$ l'ensemble des chemins de M qui ne sont préfixes d'aucun autre. Soit $Instances(P, t)$ l'ensemble de toutes les instances d'un chemin P sur t. Soit $PaternInstanceSet$ l'ensemble d'instances de chemins qui vérifient les conditions suivantes :

1. Pour tout chemin $P \in Long_M$ il existe une et une seule instance $inst \in Instances(P, t)$ dans l'ensemble $PaternInstanceSet$.
2. Pour toute instance $inst \in PaternInstanceSet$ il existe un chemin $P \in Long_M$.
3. Pour toutes instances $inst$ et $inst'$ de $PaternInstanceSet$, si $inst \in Instances(P, t)$ et $inst' \in Instances(P', t)$, alors le plus long préfixe commun à $inst$ et $inst'$ est une instance du chemin Q sur t, où Q est le plus long préfixe commun à P et P'.

Une instance d'un motif M est un n-uplet $I = (t^i, type^i, value^i)$, *i.e.*, un arbre pour lequel $type^i(t^i, p) = type(t, p)$, $value^i(t^i, p) = value(t, p)$ et t^i est une fonction $dom \rightarrow \Sigma$ où :

- $dom = \bigcup_{inst \in PaternInstanceSet} \{p \mid p$ est une position de $inst\}$
- $t^i(p) = t(p)$, $\forall p \in dom$

Puisqu'une instance de motif est distinguée seulement par les chemins commençant à la dernière position du plus long préfixe commun, dans l'exemple 1.3.1, si $\epsilon/0/0.1/0.1.0$ est dans une instance de motif alors $\epsilon/1/1.1/1.1.1/1.1.1.0$ ne peut pas être dans la même instance de motif. Ceci parce que les plus longs préfixes communs des chemins du motif sont $db/project/supplier/$ et $db/project/supplier/component$, donc ces deux instances de chemins ne peuvent pas concorder sur leur plus long préfixe commun $db/project/supplier/$.

Cette définition d'instance de motif garantit que les n-uplets de valeurs sur lesquels portent les contraintes sont constitués des bonnes valeurs (relativement à la sémantique de la contrainte). La définition suivante précise enfin les deux sortes d'égalité de nœuds utilisées dans notre cadre de spécification de contraintes.

17. Ce sont des chemins simples.

Définition 1.3.5 Egalité de valeur et égalité de nœud : Deux nœuds sont égaux par égalité de nœud s'ils sont à la même position (dans le même arbre). Deux nœuds sont égaux par égalité de valeur s'ils sont racines de deux sous-arbres isomorphes. Plus précisément, l'égalité $p =_V q$ est vérifiée si les contraintes suivantes sont vérifiées : (i) $t(p) = t(q)$, (ii) $type(p) = type(q)$, (iii) si $type(p) = data$ ou $type(p) = attribute$ alors $value(p) = value(q)$ et (iv) si $type(p) = element$ alors il existe une fonction bijective qui fait correspondre chaque position $p.i$ à une position $q.j$ telle que $p.i =_V q.j$.

Dans la figure 1.9 les nœuds aux positions 0.1.2 et 1.2.1 ne sont pas égaux en valeur, mais si la valeur en position 1.2.1.2.0 avait été 7 alors ils l'auraient été. Il est intéressant de remarquer que la définition 1.3.5 ne prend pas en compte la notion d'ordre du document. Dans la suite, lorsque l'égalité dont je parle peut aussi bien être l'égalité en valeur que l'égalité en nœud j'utiliserai le symbole E comme indice.

Avant de définir un certain nombre de types de contraintes d'intégrité dans le cadre qui vient d'être dressé, il est important de rappeler qu'un document XML T représente couramment des informations incomplètes et que cela a une incidence au niveau des contraintes d'intégrité comme au niveau des schémas (cf. section 1.1.2). Différentes propositions pour le traitement de requêtes sur des informations incomplètes, établies dans le cadre du modèle relationnel [Cod79, Lip81, Lie82, IL84, AM86, Imi89, LL99] peuvent être transposées aux documents XML [ASV01, AL02, VLL04, HT06]. Notre cadre générique d'expression des contraintes convient également pour prendre en compte cette dimension. Pour cela, dans la définition suivante, nous précisons la notion d'instance incomplète d'un chemin P sur un document $\mathcal{T}$.

Définition 1.3.6 - Instance incomplète d'un chemin P sur un document T - Soient P un chemin dans le langage PL et T un document XML. Une instance de chemin $inst$ dans T est une *instance incomplète* de P sur T ssi :
- $inst$ est une instance d'un préfixe strict P_1 de P,
- et aucune instance $inst'$ dans T de préfixe strict $inst$ n'est une instance d'un préfixe P_2 de P contenant P_1.

De là, nous étendons la notion d'instance de motif à celle *d'instance incomplète* de motif de façon analogue. Une instance de motif est incomplète si elle contient au moins une instance incomplète de chemin. La construction des n-uplets d'instance de motif est adaptée pour, à chaque instance de chemin incomplète, ajouter une valeur *null* au n-uplet. Dans [BAL09] nous montrons comment la vérification des dépendances fonctionnelles peut effectivement se traiter dans ce cadre pour les deux types de satisfaction possibles, la *satisfaction forte*[18] pour laquelle il faut un algorithme de type *chase* [MMS79, LL99] et la *satisfaction faible*[19] qui peut être traitée par l'algorithme *SatFort* [LL99].

18. Un document incomplet satisfait la XDF si et seulement si aucun remplacement d'une valeur *null* par une valeur différente de *null* ne remet en cause sa validité.

19. Lorsqu'une valeur *null* est remplacée par une valeur différente de *null*, une nouvelle validation est nécessaire.

1.3.2 Différents types de contraintes d'intégrité

Je peux maintenant donner les définitions des différentes sortes de contraintes d'intégrité existant pour XML, telles qu'elles ont été énoncées dans [Hal07, BAL09, HL10].

Pour toutes ces sortes de contraintes, il y a un chemin C qui spécifie le contexte dans lequel la contrainte s'applique. Lorsque C est le chemin vide ou le nom de l'élément racine, c'est qu'il s'agit d'une contrainte *absolue*, sinon c'est une contrainte *relative* à un contexte. Je commence par l'une des contraintes les plus étudiées, la dépendance fonctionnelle.

Définition 1.3.7 Dépendance fonctionnelle pour XML (XFD) : Etant donné un arbre XML t, une dépendance fonctionnelle (XFD) est une expression de la forme $\gamma = (C, (\{P_1\ [E_1], \ldots, P_k\ [E_k]\} \rightarrow Q\ [E]))$ où C (*le chemin définissant le contexte dans lequel la dépendance doit être vérifiée*) est un chemin partant de la racine de t et finissant sur un *nœud contexte* ; $\{P_1, \ldots, P_k\}$ est un ensemble non vide de chemins sur t et Q est un chemin sur t, les chemins P_i et Q commençant tous au nœud contexte. L'ensemble $\{P_1, \ldots, P_k\}$ est la partie gauche de la dépendance fonctionnelle (LHS), appelée determinant, tandis que Q en est la partie droite (RHS), appelée dépendant. Les symboles $E_1, \ldots, E_k, E$ représentent le type d'égalité choisi pour chacun des chemins. Lorsque les symboles E ou $E_1, \ldots, E_k$ sont omis, c'est l'égalité de valeurs qui est considérée par défaut.

Définition 1.3.8 Satisfaction de dépendance fonctionnelle : Soient $\mathcal{T}$ un document XML, $\gamma = (C, (\{P_1\ [E_1], \ldots, P_k\ [E_k]\} \rightarrow Q\ [E]))$ une dépendance fonctionnelle et M le motif de $\{C/P_1, \ldots, C/P_k, C/Q\}$. $\mathcal{T}$ satisfait γ (noté $\mathcal{T} \models \gamma$) ssi pour toutes instances I^1_M, I^2_M de M sur $\mathcal{T}$ qui concordent au moins sur leur préfixe C, il est vérifié que : $\tau^1[C/P_1, \ldots, C/P_k] =_{E_i, i \in [1 \ldots k]} \tau^2[C/P_1, \ldots, C/P_k] \Rightarrow \tau^1[C/Q] =_E \tau^2[C/Q]$ où τ^1 (resp. τ^2) est le n-uplet constitué des items aux feuilles de I^1_M (resp. I^2_M).

La notion de n-uplet est ici considérée selon la *perspective nommée* [AHV95], dans laquelle l'ordre des éléments du n-uplet n'est pas important dans la mesure où chacun est associé à son nom. Ainsi dans la définition 1.3.8, le n-uplet τ^1 est constitué à partir de I^1_M en assemblant les valeurs et les nœuds trouvés à la fin des instances de chemins de I^1_M. Chacun de ces éléments est associé au chemin correspondant dans γ, qui fait office de "nom". Notre définition des dépendances fonctionnelles est comparable à celles de [AL04, VLL04, LVL03, WT05] mais en plus général car (i) nous permettons de combiner les deux sortes d'égalité de nœud dans une même définition de dépendance, (ii) nous permettons de définir des dépendances dans un contexte précis, c'est à dire dans une partie seulement du document et (iii) nous permettons qu'il y ait plus d'un chemin dans la partie gauche de la dépendance.

Notre cadre de définition de contraintes permet évidemment de représenter les clés pour XML, telles qu'elles ont été définies dans [BDF+01].

Définition 1.3.9 Clés et clés étrangères pour XML (XKey et XFK) [BDF+01] Une clé est une expression $(C, (T, \{P_1, \ldots, P_k\}))$ dans laquelle le chemin C est le *chemin contexte*, le chemin T est le *chemin cible* et les chemins $P_1, \ldots, P_k$ sont les *chemins clé*. Une clé étrangère est une expression $(C, (T', \{P'_1, \ldots, P'_k\}) \subseteq (T, \{P_1, \ldots, P_k\}))$ où $(C, (T, \{P_1, \ldots, P_k\}))$ est une clé K et $\{P'_1, \ldots, P'_k\}$ sont les *chemins clé étrangère*.

Définition 1.3.10 Satisfaction de clé [BAL09] : Soient $\gamma = (C, (T, \{P_1, \ldots, P_k\}))$ une clé et M le motif de $\{C/T/P_1, \ldots, C/T/P_k\}$. $\mathcal{T}$ satisfait γ (noté $T \models \gamma$) ssi (i) pour toute instance i de C/T, il existe une et seulement une instance I_M de M correspondant au préfixe commun i et (ii) $\forall I^1_M, I^2_M$, instances de M qui concordent au moins sur C, il est vérifié que : $\forall i \in [1 \ldots k], \tau^1[C/T/P_i] \neq \perp$ et $\tau^1[C/T/P_i] =_v \tau^2[C/T/P_i] \Rightarrow \tau^1[C/T] = t^2[C/T]$ où τ^1 (resp. τ^2) est le n-uplet constitué des items se trouvant aux feuilles de I^1_M (resp. I^2_M).

Une clé est vérifiée si tous les n-uplets de valeurs correspondant au motif de $\{P_1, \ldots, P_k\}$ (dont tous les chemins commencent à l'un des nœuds cibles spécifés par T) sont uniques dans le contexte de chaque sous-arbre enraciné à un nœud "contexte".

Exemple 1.3.2 Le document en figure 1.9 satisfait la clé $(db/project, (supplier, \{@sname\}))$, ainsi que les dépendances fonctionnelles suivantes :

$XFD_1 : (db, (\ \{/project/pname\} \rightarrow /project\ [N]\ ))$
Les noms de projet sont uniques et identifient un projet. Le contexte étant *db*, la racine du document, cette dépendance doit être vérifiée dans tout le document.

$XFD_2 : (db, (\ \{/project/pname\} \rightarrow /project))$
Les sous-arbres enracinés en un nœud "project" qui ont un nom identique sont isomorphes.

$XFD_3 : (db/project, (\{/supplier/@sname, /supplier/component/@cname\} \rightarrow /supplier/component/quantity\ ))$
Dans le contexte d'un projet, pour un couple (nom de produit, nom de fournisseur) il n'existe qu'une seule quantité. □

Sur le même principe que pour une clé, une clé étrangère est vérifiée si tous les n-uplets de valeurs correspondant au motif de $\{P'_1, \ldots, P'_k\}$ existent comme n-uplets de valeurs de clé, dans le contexte de chaque sous-arbre enraciné à un nœud contexte. Les clés étrangères sont des cas particuliers de dépendances plus générales, les dépendances d'inclusion.

Définition 1.3.11 Dépendances d'inclusion pour XML (XID) [BAL09] : Une dépendance d'inclusion est une expression de la forme $(C, (\{P_1, \ldots, P_k\} \subseteq \{Q_1, \ldots, Q_k\}))$ dans laquelle tous les chemins C/P_i ($\forall i \in [1 \ldots k]$) et C/Q_i sont des chemins partant de la racine du document considéré. Le chemin C est le chemin contexte.

Définition 1.3.12 Satisfaction de dépendances d'inclusion [BAL09] : Soient $\mathcal{T}$ un document XML, $\gamma = (C, (\{P_1, \ldots, P_k\} \subseteq \{Q_1, \ldots, Q_k\}))$ une dépendance d'inclusion et M^1 et M^2 les motifs de $\{C/P_1, \ldots, C/P_k\}$ et (resp.) de $\{C/Q_1, \ldots, C/Q_k\}$. $\mathcal{T}$ satisfait γ (noté $T \models \gamma$) ssi pour toute instance I_{M^1} de M^1 il existe une instance I_{M^2} de M^2 qui concorde avec I_{M^1} au moins sur C et $\forall i \in [1, \ldots, k]$, $\tau^1[C/P_i] \subseteq \tau^2[C/Q_i]$, où τ^1 (resp. τ^2) est le n-uplet constitué des items se trouvant aux feuilles de I_{M^1} (resp. I_{M^2}).

Notre cadre générique permet également d'exprimer les *contraintes d'inverse* pour XML (XIC) définies dans [BFSW01] et également utilisées dans les bases de données objet, qui reposent sur une double inclusion : une contrainte d'inverse est une expression de la forme

$(C, (\{P_1, \ldots, P_k\} = \{Q_1, \ldots, Q_k\}))$. Elle exprime que, dans le contexte défini par le chemin C, les n-uplets d'items trouvés aux feuilles des instances du motif de $\{C/P_1, \ldots, C/P_k\}$ sont égaux (en valeur) à ceux trouvés aux feuilles des instances du motif de $\{C/Q_1, \ldots, C/Q_k\}$.

Il est également possible d'exprimer les *contraintes numériques* pour XML définies dans [HL10], comme le précise la définition suivante.

Définition 1.3.13 Contraintes numériques pour XML (XNC) [HL10] : Une contrainte numérique est une expression de la forme $card(Q, (Q', \{Q_1, \ldots, Q_k\})) = (min, max)$ où Q, Q', $Q_1, \ldots, Q_k$ sont des expressions telles que Q/Q', $Q/Q'/Q_1, \ldots, Q/Q'/Q_k$ sont des chemins valides, $min \in \mathbb{N}$, $max \in \mathbb{N}$, et $min \leq max$.

Définition 1.3.14 Satisfaction de contraintes numériques [HL10] : Soient $\varphi = card(Q, (Q', \{Q_1, \ldots, Q_k\})) = (min, max)$ une contrainte numérique et $\mathcal{T}$ un document XML. $\mathcal{T}$ satisfait φ, (noté $\mathcal{T} \models \varphi$) ssi pour tout $q \in \epsilon\|Q\|$ [20], pour tout $q'_0 \in q\|Q'\|$ tel que pour tout $x_1, \ldots, x_k$ où $x_i \in q'_0\|Q_i\|$ (pour $i \in [1 \ldots k]$), il est vérifié que $min \leq \bar{S} \leq max$, où $\bar{S}$ est la cardinalité de l'ensemble fini S, et $S = \{q' \in q\|Q'\| \mid \exists y_1, \ldots, y_k. \forall i \in [1 \ldots k]. y_i \in q'\|Q_i\| \wedge x_i =_v y_i\}$.

Exemple 1.3.3 La contrainte numérique $XNC_1 : card(db/project, (supplier, \{//@cname\})) = (1, 2)$ indique que dans tout projet chaque composant provient au minimum d'un et au maximum de deux fournisseurs. □

Les *clés numériques* (XNK) sont un cas particulier de contrainte numérique, dans lequel min est fixé à 1. Une clé numérique pour XML est une expression de la forme $card(Q, (Q', \{Q_1, \ldots, Q_k\})) \leq max$. On peut noter que lorsque $max = 1$ cela revient à une clé XML.

1.4 Cadre générique de validation pour les contraintes d'intégrité

Au-delà d'un cadre commun de spécification des contraintes d'intégrité pour XML, notre contribution la plus importante à ce domaine est la définition d'un cadre générique de validation effective de ces contraintes dans un document XML, que ce soit complètement ou incrémentalement lors de mises-à-jour du document. En effet, comme pour les schémas, la validation de contraintes d'intégrité comprend naturellement deux volets : d'abord concevoir un algorithme pour la validation initiale de tout le document, puis déterminer les informations et actions nécessaires pour réaliser le minimum possible de tests afin de vérifier qu'une mise-à-jour n'introduit pas d'invalidité.

1.4.1 Validation initiale

Le cadre que nous avons mis au point pour la validation initiale repose sur la notion de grammaire attribuée [ASU88], également utilisée pour XML par [Nev99]. Nous définissons

20. $\nu\|Q\|$ denote l'ensemble de nœuds de T que l'on peut atteindre depuis ν en suivant une instance de Q.

une grammaire attribuée générique qui représente un document XML bien formé et nous "compilons" chaque type de contrainte vers un ensemble d'attributs de cette grammaire, qui vont recevoir leur valeur au cours du parcours linéaire du document, c'est-à-dire *en une seule traversée complète de l'arbre, quelque soit le nombre et le type des contraintes à vérifier.*

Une grammaire attribuée est une grammaire hors contexte étendue par des attributs [ASU88]. Chaque symbole de la grammaire est lié à un ensemble fini d'attributs, éventuellement vide. Chaque attribut a un domaine de valeurs. La valeur de chaque attribut à chaque nœud de l'arbre syntaxique est définie par des règles. Les attributs associés à un symbole sont de deux types :

- synthétisés : la règle de calcul utilise les valeurs des attributs associés aux fils pour donner une valeur à l'attribut du nœud courant ;
- hérités : la règle de calcul utilise les valeurs des attributs associés aux frères gauches et au père pour donner une valeur à l'attribut du nœud courant.

La définition suivante précise ces notions.

Définition 1.4.1 Grammaire attribuée et règles de calcul des attributs [ASU88] : Une grammaire attribuée est un triplet $GA = (G, A, F)$ où : $G = (V_N, V_T, P, B)$ est une grammaire hors contexte (chaque production dans P est de la forme $p : X_0 \to X_1 \dots X_n$ où $X_0 \in V_N$ et $X_i \in (V_N \cup V_T)^*$, $i \in [1 \dots n]$), A est l'ensemble des attributs et F est l'ensemble de règles de calcul. Précisément, ces ensembles sont constitués de la manière suivante : pour tout $X \in V_N \cup V_T$, $A(X)$ est l'union disjointe de $S(X)$, l'ensemble des attributs synthétisés de X et $I(X)$, l'ensemble des attributs hérités de X. $X.a$ dénote l'attribut a de $A(X)$. De même, l'ensemble des attributs impliqués dans une règle de production p est noté $A(p) = \{X_i.a \mid a \in A(X_i), i \in [0 \dots n]\}$. L'ensemble $F(p)$ contient les règles de calcul qui traitent l'ensemble des attributs $A(p)$. Ces règles de calcul sont de la forme $b := f(c_1, c_2, ..., c_k)$, où f est une fonction et (i) ou bien b est un attribut synthétisé associé à X_0 et $c_1, c_2, ..., c_k$ sont des attributs associés aux symboles non terminaux X_i, (ii) ou bien b est un attribut hérité associé à un symbole X_i et $c_1, c_2, ..., c_k$ sont des attributs associés à X_0 ou à des symboles non terminaux X_j, $j \in [1, \dots, i]$.

Les grammaires attribuées ont été conçues de façon à ce que l'analyse sémantique d'une phrase soit accomplie par un ensemble d'actions associées à chaque règle de production, ces actions calculant les valeurs des différents attributs associés aux nœuds de l'arbre syntaxique de la phrase. Dans le contexte de la validation de contraintes d'intégrité pour XML, l'arbre est un arbre XML, aussi la grammaire générique G représente un arbre XML avec trois règles, l'une pour le nœud racine, l'autre pour les nœuds contenant une valeur (incluant les attributs) et la troisième pour les autres nœuds, c'est-à-dire, en considérant que $\alpha_1 \dots \alpha_m$ dénote la séquence des nœuds fils (attributs ou éléments) d'un élément A ou de l'élément racine $ROOT$:

- Règle pour l'élément racine : $ROOT \to \alpha_1 \dots \alpha_m$, $m \in \mathbb{N}$.
- Règle pour un élément quelconque A : $A \to \alpha_1 \dots \alpha_m$, $m \in \mathbb{N}^*$.
- Règle pour un élément contenant une valeur et pour un attribut : $A \to data$.

Notons qu'il aurait été possible de considérer les règles d'un schéma pour XML comme règles de grammaire, à enrichir des attributs et règles de calcul. Mais comme je l'ai déjà

précisé en section 1.3, notre cadre générique de spécification de contraintes d'intégrité ne présuppose pas l'existence d'un schéma (même s'il reste valable en présence d'un schéma). Les règles de production de la grammaire G sont donc on ne peut plus générales et ce sont les attributs (ensemble A) et les règles de calcul (ensemble F) qui vont assurer la validation de chaque contrainte.

Etant donné que pour cette grammaire attribuée générique l'arbre syntaxique correspond à l'arbre XML à valider, la validation est réalisée au cours d'un simple parcours de l'arbre XML. Dans le sens descendant (à la rencontre des balises ouvrantes dans le document) le processus de validation utilise l'attribut hérité appelé $conf$ pour associer à chaque nœud de l'arbre XML le rôle qu'il joue par rapport à chacune des contraintes vérifiées. Dans le sens ascendant (rencontre de balise fermante dans le document), le processus de validation utilise les attributs synthétisés pour former les n-uplets de valeurs sur lesquels les tests devront être opérés pour vérifier chacune des contraintes. L'algorithme 1 exprime cette démarche générique de validation d'un document par rapport à une contrainte d'intégrité.

Algorithme 1 - Validation d'une contrainte d'intégrité C

Entrées :

(i) t : arbre XML
(iii) C : la contrainte d'intégrité
(iv) MEF_C : ensemble des machines à états finis correspondant à la contrainte C

Sortie :

Si t est valide par rapport à C alors $true$ sinon $false$

Variables locales :

(i) $CONF$: stocke les valeurs des attributs hérités
(ii) $SYNT$: stocke les valeurs des attributs synthétisés

1. Initialize $CONF_\epsilon$ (using MEF_C)
2. **for each** *event*[21] v *in* t
3. **switch** (v) **do**
4. **case** *open tag in position* p
5. Compute $CONF_p$ using MEF_C
6. **case** *closing tag in position* p
7. Compute $SYNT_p$ using $CONF_p$ and $SYNT_{p.i}$, $0 \leq i \leq t^-_{|_p}$[22]
8. **case** *attribute* or *value* in position p
9. Compute $SYNT_p$ using $CONF_p$ and $value(p)$
10. **end for**
11. **if** $\exists false \in SYNT_\epsilon.c$ **then** return false **else** return true □

Le rôle que joue un nœud de l'arbre XML par rapport à une contrainte est déterminé par un ensemble de machines à états finis (MEF), qui varient en fonction du type de contrainte. Cependant comme toutes comprennent un chemin C pour spécifier le contexte, il y a toujours un automate à états finis A_C pour représenter le chemin C qui doit être

21. événement SAX : balise ouvrante, attribut, valeur, balise fermante, etc.
22. Nombre de fils directs du nœud racine du sous-arbre enraciné en position p : cf. définitions 1.1.1 et 1.1.2.

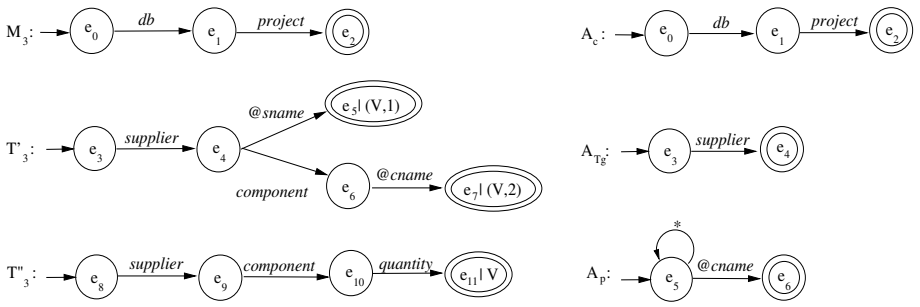


FIGURE 1.10 – Automates et transducteurs pour XFD_3 et XNC_1.

suivi en partant de la racine du document jusqu'aux nœuds contextes. Les nœuds contexte sont les points où les contraintes doivent être vérifiées. Selon le type de la contrainte, les autres chemins qui la spécifient sont associés à différents types de machines à états finis, parfois simples automates comme pour les clés, parfois transducteurs comme pour les dépendances fonctionnelles. Ces machines à états finis correspondent aux définitions classiques. Un automate à états finis est un quintuplet $A = (\Theta, V, \Delta, e, F)$ où Θ est l'ensemble fini des états, V est l'alphabet, $e \in \Theta$ est l'état initial, $F \subseteq \Theta$ est l'ensemble des états finaux et $\Delta : \Theta \times V \rightarrow \Theta$ est la fonction de transition. Pour une dépendance fonctionnelle ou une dépendance d'inclusion on utilise un transducteur, sextuplet $A = (\Theta, V, \Gamma, \Delta, e, F, \lambda)$ où (i) $(\Theta, V, \Delta, e, F)$ est un automate à états finis, (ii) Γ est l'alphabet de sortie et (iii) λ est une fonction de F dans Γ qui spécifie la sortie associée à chaque état final. Dans le cadre des contraintes d'intégrité, l'alphabet V est toujours l'ensemble Σ de la définition 1.3.1, l'alphabet Γ pour les transducteurs est construit à partir des symboles $\{V, N\}$ qui dénotent les deux sortes d'égalité de nœuds (définition 1.3.5). L'algorithme de validation associe à chaque nœud de l'arbre XML un attribut $conf_i$ (lignes 1 et 5 de l'algorithme 1), pour chaque contrainte γ_i à valider. La valeur de $conf_i$ est un ensemble de *configurations d'automate*. Une configuration d'automate est dénotée $A.s$, où A est l'automate et s est un de ses états.

Par exemple la figure 1.10 présente les machines à états finis qui correspondent à la dépendance fonctionnelle
$XFD_3 : (db/project, (\{/supplier/@sname, /supplier/component/@cname\}$
$\rightarrow /supplier/component/quantity\ ))$
et à la contrainte numérique
$XNC_1 : card(db/project, (supplier, \{//@cname\})) = (1, 2)$
données dans les exemples 1.3.2 et (resp.) 1.3.3 en section 1.3. Pour XFD_3 , nous savons d'après la définition 1.3.7 que les chemins C, P_i et Q ($i \in [1 \ldots k]$) représentent respectivement le contexte, la partie déterminant et la partie dépendant de la contrainte. Ils sont représentés par les automates suivants (cf. figure 1.10) :

- L'automate contexte A_C, noté ici $M_3 = (\Theta, \Sigma, \Delta, e, F)$, pour C.
- Le transducteur déterminant $T'_3 = (\Theta', \Sigma, \Gamma', \Delta', e', F', \lambda')$ pour les chemins P_i ($i \in [1 \ldots k]$). L'ensemble $\Gamma' = \{V, N\} \times \mathbb{N}^*$ où V et N sont les sortes d'égalité de nœud et i dénote le chemin P_i, $i \in [1 \ldots k]$ dans lequel le symbole est utilisé.
- Le transducteur dépendant $T''_3 = (\Theta'', \Sigma, \Gamma'', \Delta'', e'', F'', \lambda'')$ pour Q, où $\Gamma'' = \{V, N\}$.

La table 1.1 synthétise pour chaque type de contrainte d'intégrité introduit en section 1.3 les machines à états finis, les attributs hérités et les attributs synthétisés utilisés pour représenter la contrainte dans la grammaire attribuée, et donc dans l'algorithme 1. Quelque soit le type de contrainte les attributs suivants sont utilisés : $conf$, que j'ai déjà évoqué et qui représente le rôle que joue le nœud par rapport à la contrainte, c, qui représente le résultat de la validation et d'autres attributs synthétisés dans lesquels les valeurs utiles sont rassemblées pour construire les n-uplets des instances de motifs ($inters$, ds_j, dc ou ds_j^R pour les dépendances fonctionnelles (XFD), les dépendances d'inclusion (XID) et d'inverse (XIC) et f et tg pour les clés (XKey), les clés étrangères (XFK), les contraintes numériques (XNC) et les clés numériques (XNK)).

Table 1.1 – Représentation des différents types de contraintes d'intégrité par les attributs de la grammaire attribuée G.

Contrainte	Expression	MEF	Attributs
XFD	$(C, (\{P_1\ [E_1], \ldots, P_k\ [E_k]\} \rightarrow Q\ [E]))$	A_C, T et T'	Hérité : $conf$ Synth. : $c, inters, ds_j, dc$
XID	$(C, (\{P_1^R, \ldots, P_k^R\} \subseteq P_1, \ldots, P_k))$	A_C, T et T'	Hérité : $conf$ Synth. : $c, inters, ds_j, ds_j^R$
XIC	$(C, (\{P_1^R, \ldots, P_k^R\} = P_1, \ldots, P_k))$	A_C, T et T'	Hérité : $conf$ Synth. : $c, inters, ds_j, ds_j^R$
XKey	$(C, (T_g, \{P_1, \ldots, P_k\}))$	A_C, A_{Tg} et A_P	Hérité : $conf$ Synth. : c, tg et f
XFK	$(C, (T_g^R, \{P_1^R, \ldots, P_k^R\}) \subseteq (T_g, \{P_1, \ldots, P_k\}))$	A_C, A_{Tg}^R, A_P^R	Hérité : $conf$ Synth. : c, tg et f
XNC	$card(C, (Tg, \{P_1, \ldots, P_k\})) = (min, max)$	A_C, A_{Tg} et A_P	Hérité : $conf$ Synth. : c, tg et f
XNK	$card(C, (Tg, \{P_1, \ldots, P_k\})) \leq max$	A_C, A_{Tg} et A_P	Hérité : $conf$ Synth. : c, tg et f

L'aspect *générique* de notre contribution repose donc sur le fait que toute contrainte d'intégrité peut être vérifiée par la même grammaire, seuls les attributs et les règles de calcul sont propres à chaque type de contrainte. Il est intéressant de noter que la structure générale de l'algorithme 1 est exactement identique à celle de l'algorithme général de validation d'un document par rapport à un schéma détaillé dans [BCF+07], ce qui renforce encore l'aspect générique : en un parcours du document l'ensemble des contraintes de structure et de données peuvent être vérifiées.

Le détail des calculs des attributs peut être trouvé dans [BCF+07, Lim07, BHdL11] pour les clés, les clés étrangères, les dépendances fonctionnelles et les dépendances d'inclusion. J'en donne simplement ici une intuition pour la validation de la dépendance fonctionnelle et

celle de la contrainte numérique dont les machines à états finis sont données en figure 1.10. Dans la phase descendante, les attributs hérité $conf_i$ reçoivent leurs valeurs grâce aux machines à états finis qui représentent les contraintes. Pour certains nœuds de t la valeur de $conf_i$ est l'ensemble vide, ce qui indique que ces nœuds ne sont sur le chemin d'aucune partie de la contrainte γ_i. La figure 1.11 montre les valeurs de l'attribut $conf_i$ pour XFD_3 et XNC_1, commentées dans l'exemple suivant.

Exemple 1.4.1 Soient $conf_1$ et $conf_2$ les attributs hérités pour XFD_3 et XNC_1. La contrainte XFD_3 est representée par M_3, T'_3 et T''_3 de la figure 1.10 et XNC_1 est représentée par A_C, A_{Tg} et A_P également en figure 1.10. Les valeurs de $conf_1$ et $conf_2$ sont calculées depuis la racine vers les feuilles. A la racine, ces valeurs sont, respectivement, $\{M_3.e_1\}$ et $\{A_C.e_1\}$. On peut constater que le nœud en position 0.0 ne joue aucun rôle dans les deux contraintes considérées. Par contre le nœud en position 0.1 a $conf_1 = \{T'_3.e_4, T"_3.e_9\}$ et $conf_2 = \{A_{Tg}.e_4\}$, ce qui indique qu'il est un nœud cible de la contrainte XNC_1. Son père (en position 0) est un nœud contexte à la fois pour XFD_3 et XNC_1, car il a $conf_1 = \{M_3.e_2\}$, c'est-à-dire un état final de M_3, et $conf_2 = \{A_C.e_2\}$, c'est-à-dire un état final de A_C. □

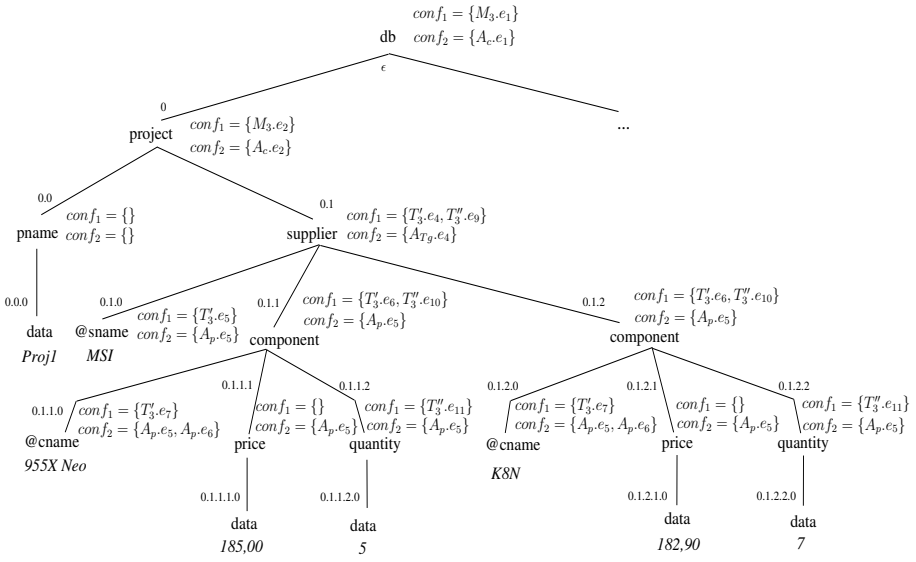


FIGURE 1.11 – Attributs hérités $conf_1$ et $conf_2$ pour XFD_3 et (resp.) XNC_1.

Une fois atteintes les feuilles, le processus de validation utilise les attributs synthétisés pour rassembler progressivement les bonnes valeurs au sein des n-uplets, en remontant jusqu'aux nœuds contextes où ces n-uplets de valeurs doivent être testés. Pour une dépendance fonctionnelle il utilise $k + 3$ attributs synthétisés, k étant le nombre de chemins de la partie déterminant (cf. définition 1.3.7), ces attributs synthétisés étant dénotés par c, $inters$, dc et ds_j $(1 \leq j \leq k)$. L'attribut c est utilisé simplement pour remonter le résultat de la validation ($true$ ou $false$) depuis les nœuds contexte jusqu'à la racine. L'attribut $inters$ remonte les items qui sont dans l'intersection des chemins de motif pour la partie déterminant et pour la partie dépendant. Ces items peuvent être des valeurs $data$ ou des positions (nœuds), selon que E et E_j sont V ou N dans la définition de la dépendance. Ces items sont d'abord récoltés (depuis les feuilles pour les $data$) par les attributs ds_j et dc .

Pour les dépendances d'inclusion l'idée est la même mais cette fois il faut récolter des n-uplets d'items dans les attributs ds_j^R $(1 \leq j \leq k)$ (à la place de simplement un item dans dc), ceci afin de tester s'ils correspondent aux valeurs récoltées dans ds_j. Pour les clés, les clés étrangères et les contraintes ou clés numériques, le processus utilise trois attributs, c, tg et f, pour (respectivement) les résultats booléens (entre les nœuds contexte et la racine), les n-uplets complets entre les nœuds cibles et leur contexte et les valeurs constituant progressivement les n-uplets entre les feuilles et les nœuds cibles. A chaque nœud ces attributs reçoivent des valeurs selon le rôle qu'ils jouent dans la contrainte (attribut $conf_i$) et selon les valeurs des attributs c, tg et f de leurs nœuds fils.

Le challenge concernant les calculs des attributs synthétisés pour chaque sorte de contrainte est de faire en sorte de constituer précisément les n-uplets des instances de motifs telles que définies en section 1.3. Les algorithmes correspondants sont détaillés dans la thèse de Maria Adriana Vidigal de Lima [Lim07] et dans nos publications [BCF+07, BHdL11, BFL12]. Je donne simplement ici une intuition sur le fonctionnement de ces calculs, au travers de nos deux exemples récurrents, illustrés par la figure 1.12.

Exemple 1.4.2 La figure 1.12 montre le calcul des attributs c_1, *inters*, ds_j et dc pour XFD_3, ainsi que f, tg et c_2 pour XNC_1. Etant donnée la partie déterminant de XFD_3, les attributs ds_j contiennent les valeurs de @*fname* (nom du fournisseur) et @*cname* (nom du composant). Pour la partie dépendant, dc prend la valeur de *quantity*. Les attributs ds_j et dc remontent leurs valeurs jusqu'au premier nœud intersection (des chemins du motif), en l'occurrence *component* en position 0.1.1 : il y a là $ds_1 = < \epsilon >$, $ds_2 =< 955X\ Neo >$ et $dc =<5>$. L'attribut *inters* reçoit alors un couple contenant d'une part le n-uplet des ds_j et d'autre part dc, soit ici : $inters = \{< <\epsilon, 955X\ Neo>, 5 >\}$. De la même manière, au nœud *component* en position 0.1.2 l'attribut *inters* reçoit la valeur $\{< <\epsilon, K8N>, 7 >\}$. A l'intersection suivante, c'est-à-dire au nœud *supplier* en position 0.1, le n-uplet calculé de la même façon à partir de l'attribut ds_1 du nœud en position 0.1.0 (*@fname*) est : $< <MSI,\epsilon>, \epsilon>$. L'attribut *inters* de ce nœud *supplier* reçoit alors la jointure entre ce nouveau n-uplet et ceux des attributs *inters* de ses nœuds fils : $< < <MSI, 955XNeo>,5>, < <MSI, K8N>,7> >$. Dans le nœud contexte, ici *project* en position 0, la dépendance est vérifiée conformément à la définition 1.3.8 et l'attribut c_1 prend donc la valeur *true*, qui est remontée jusqu'à la racine, où il est vérifié que toutes les valeurs des attributs c_1 remontées de tous les nœuds contexte sont *true*.

Pour XNC_1, les valeurs de @*cname* sont sélectionnées dans l'attribut f et remontées aux nœuds cible, l'union des f devient le premier item de tg. Ainsi en position 0.1 dans la figure 1.12, on a $tg = \{\{< 995X\ Neo >, < K8N >\}\}$. Au niveau des nœuds contexte (encore *project* pour XNC_1), l'union TG de tous les attributs tg des nœuds fils est réalisée. TG contient donc un ensemble d'ensembles de n-uplets, chaque ensemble dans TG correspondant à un nœud cible. Dans notre exemple, il n'y a qu'un seul nœud cible (*supplier*) et chaque n-uplet n'a qu'un seul élément, le nom d'un composant. Pour chaque composant, par exemple $995X\ Neo$, il faut alors compter le nombre d'ensembles dans TG qui contiennent ce même nom de composant (*e.g.* $995XNeo$) et vérifier que ce nombre est toujours entre les valeurs *min* et *max* spécifiées dans la contrainte numérique. Dans notre exemple, , le processus de validation vérifie ainsi que, pour le projet en position 0, chaque nom de composant (dans notre exemple soit $995XNeo$, soit $K8N$) est fourni par au moins 1 et au plus 2 fournisseurs et c'est bien le cas puisqu'il n'y a qu'un seul ensemble dans TG. □

La validation initiale synthétisée par l'algorithme 1 est linéaire sur la taille du document. Selon la sorte de contrainte, la complexité des calculs des valeurs des attributs à chaque nœud est différente mais elle reste toujours polynomiale sur le nombre de valeurs locales considérées. Les implémentations qui ont été menées utilisent systématiquement des tables

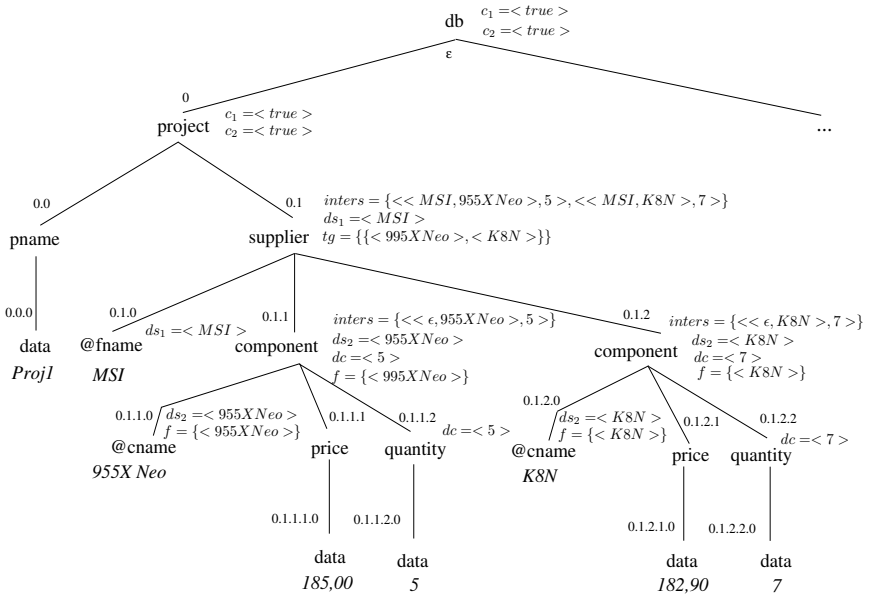


FIGURE 1.12 – Attributs synthétisés c_1, $inters$, ds_j et dc pour XFD_3, et c_2, tg et f pour XNC_1.

de hachage pour optimiser les comparaisons de valeurs. Les expérimentations montrent des temps d'exécution très corrects, par exemple un peu moins de 7 secondes pour 15000 n-uplets correspondant à 3 dépendances fonctionnelles différentes (ce n'est pas tant la taille du document que le nombre d'instances des dépendances qu'il contient qui influe sur le temps de calcul). Bien qu'il soit difficile, comme pour les algorithmes de correction de document par rapport à un schéma, de comparer précisément avec les propositions publiées par ailleurs (dont les expérimentations sont difficilement reproductibles), ces performances sont au moins aussi bonnes que celles des autres approches, qui traitent toujours un seul type de contrainte.

Les autres algorithmes de validation publiés ne sont pas nombreux : des implémentations de vérification de dépendances fonctionnelles, fondés sur des tables de hachage comme les nôtres, sont présentés dans [LLL02, VL05, SL09]. Leur complexité est donnée comme linéaire sur le nombre de chemins composant la contrainte et sur le nombre de n-uplet instances de la contraintes existant dans le document. Des algorithmes de validation de clés et clés étrangères ont également été proposés, par exemple dans [BBG+02, BCF+07, CDZ02]. Dans [BBG+02] les gains obtenus par une version incrémentale de la validation suite à une mise-à-jour sont mis en avant, mais seulement pour une clé et pour une seule contrainte. De fait, non seulement la complexité théorique et les performances de nos prototypes montrent que notre cadre de validation est à la hauteur de ces diverses propositions, mais c'est surtout son aspect générique qui le rend supérieur. D'autant plus que nous parvenons à étendre cette généricité à la validation incrémentale des contraintes d'intégrité.

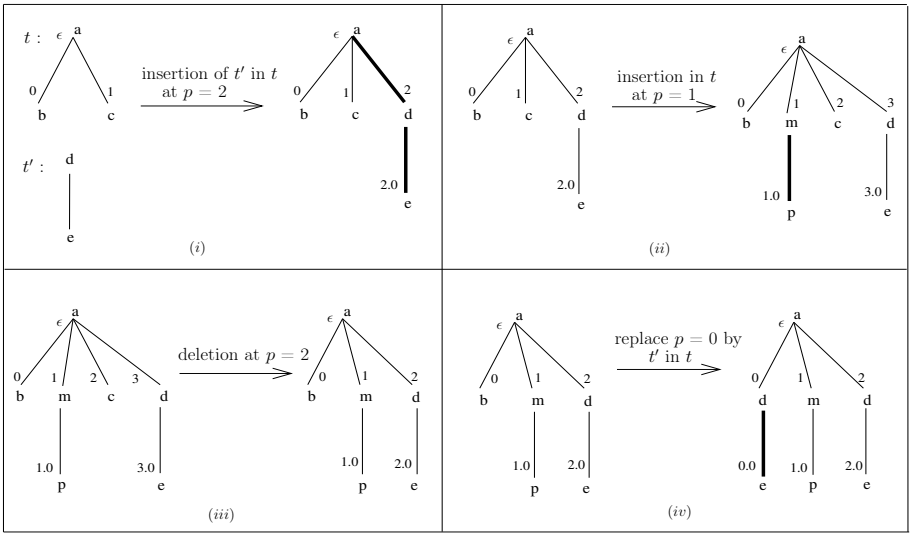


Figure 1.13 – Exemples de dérivations par une unique opération de mise-à-jour sur t. (i) Insertion en une position frontière (cf. définition 1.1.9). (ii) Insertion en une position p de t. (iii) Suppression d'un sous-arbre. (iv) Remplacement d'un sous-arbre par un autre.

1.4.2 Validation incrémentale

Une opération de mise-à-jour sur un arbre t est un triplet $\langle p, op, t' \rangle$ où p est la position de mise-à-jour, op est le type de mise à jour ($insert$, $remove$, $replace$) et t' est un arbre XML étendu. Les opérations correspondant à $insert$, $remove$, $replace$ sont celles décrites en définition 1.1.12. La figure 1.13 illustre l'application de chacune de ces opérations.

Dans notre proposition de validation incrémentale nous considérons une séquence de telles opérations de mises-à-jour, stockée dans une structure que nous appelons *UpdateList*. Ainsi nous traitons des mises-à-jour multiples sous la forme classique d'une *transaction*, en suivant les principes décrits dans [BBFV05, SHS04] où la validité d'un document n'est décidée qu'après avoir considéré toutes les mises-à-jour d'une transaction donnée. Plus précisément, nous considérons des séquences de mises à jour *non-contradictoires*[23] telles que définies dans [Hal07, BFL12].

Nous considérons que pour cela un prétraitement est fait sur les mises-à-jour avant application de notre algorithme de validation incrémentale, qui permet aussi d'ordonner les opérations selon les positions impliquées, pour que ces positions soient dans l'ordre du document[24]. Pour ce qui est de la validation incrémentale en elle-même, l'ordre entre les opérations n'influe pas sur le résultat car nous utilisons des structures de stockage auxiliaires qui permettent d'admettre temporairement des cas de non validité en attendant la fin de la transaction.

23. Compatibles les unes avec les autres : (1) il ne peut y avoir qu'une unique opération si sa position est la racine, (2) les opérations de suppression et remplacement ne peuvent porter sur les mêmes positions, (3) ni sur leurs descendants.

24. Il est démontré dans [Hal07] que des ordres différents d'opérations de mises-à-jour permettent de dériver des arbres toujours isomorphes les-uns aux autres et que, s'il n'y a pas d'opération d'insertion alors on obtient des arbres identiques.

Ces structures de données temporaires sont le cœur de la validation incrémentale, elles doivent avoir été initialisées lors de la validation initiale du document par rapport aux contraintes contenues dans IC. Précisément, à la fin de la validation initiale nous associons au document, pour chaque contrainte, un n-uplet que nous appelons *n-uplet de validation* de cette contrainte. Il rassemble des informations sur les valeurs qui sont concernées par la contrainte, il est donc une sorte de résumé du document concernant cette contrainte. Lors des mises-à-jour, les valeurs sont recherchées dans ce résumé et non pas dans le document complet. Un n-uplet de validation d'une contrainte C a la structure générale suivante : $\langle\langle l_1, \ldots, l_n\rangle, Struct_1, Struct_2 \ldots\rangle$, où :

- $\langle l_1, \ldots, l_n\rangle$ est un n-uplet de n-uplets contenant les informations synthétisées, n étant le nombre de contextes présents dans le document. Par exemple pour une dépendance fonctionnelle, chaque n-uplet l_i est de la forme $\langle c, inters, ds_i, dc\rangle$ et contient les valeurs des attributs synthétisés au niveau du i^{eme} contexte tels qu'ils étaient à la fin de la validation précédente. Pour une clé, l_i est de la forme $\langle c, tg, k\rangle$, ce qui correspond aux attributs synthétisés des clés et clés étrangères (cf. table 1.1).
- $Struct_1, Struct_2, \ldots$ sont des structures auxiliaires, dont le nombre et le format dépendent du type de contraintes. Par exemple pour une dépendance fonctionnelle il y a *XFDValIndex*, index à trois niveaux et *XFDInterPos*, table de hachage pour les positions d'intersection de XFD, illustrés en figure 1.14. Pour une dépendance fonctionnelle XFD donnée, les couples $(\langle ds_1 \ldots ds_k\rangle, dc)$ stockés dans *XFDValIndex* pour chaque contexte c_i sont les n-uplets (*items du determinant*, *item du dépendant*) trouvés dans t sous le nœud en position c_i. Pour une clé c'est un index des clés, de leur position (et contexte) et des références associées décrit dans [ABH+04, BCF+07, BAL09], que nous avons appelé *keyTree*.

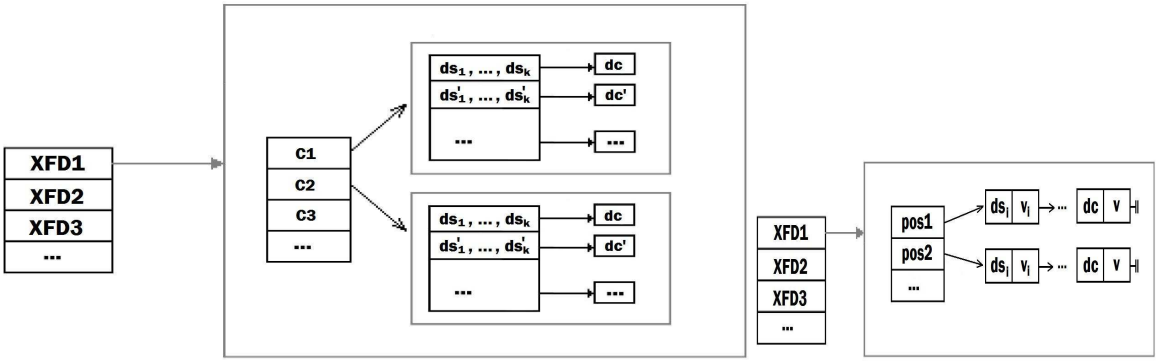


FIGURE 1.14 – Structures auxiliaires pour les dépendances fonctionnelles : à droite *XFDValIndex* et à gauche *XFDInterPos*.

Les mises-à-jour peuvent comporter des insertions ou des remplacements de sous-arbres. Dans ce cas, comme pour les schémas, la validation incrémentale doit valider *localement* les sous-arbres qu'il faut introduire dans l'arbre XML t donné, par rapport à chaque contrainte de l'ensemble IC. Il est également nécessaire pour la validation incrémentale d'avoir des informations sur la validité *locale* des sous-arbres à supprimer. La définition suivante précise la notion de validité locale pour ce qui concerne les contraintes d'intégrité.

Définition 1.4.2 Validité locale par rapport à des contraintes d'intégrité : Soient t' un arbre XML et IC un ensemble de contraintes d'intégrité, t' est localement valide par rapport à IC si le résultat de sa validation est un n-uplet $\langle\langle l_1, \ldots, l_n\rangle, Struct_1, Struct_2, \ldots\rangle$ qui respecte les conditions suivantes pour tout n-uplet l_j $(1 \leq j \leq n)$:

(i) Si la racine de t' est une position pour laquelle un attribut a_j inclus dans l_j contient des n-uplets de valeurs [25] qui représentent la contrainte vérifiée, alors chaque n-uplet de a_j, a exactement pour longueur m_j, le nombre d'items composant la contrainte.

(ii) Si la racine de t' est la j^{eme} position contexte pour une contrainte C de IC, ou une position sur le chemin du contexte de C, alors l'attribut c_j de l_j contient la valeur *true*.

La validation incrémentale générique (à adapter pour chaque sorte de contrainte) consiste en un parcours unique du document, au cours duquel des actions sont déclenchées uniquement aux nœuds impactés par une mise-à-jour, formalisé dans l'algorithme 2. Dans ce parcours, les violations temporaires de chaque sorte de contrainte sont retenues dans une structure auxiliaire, par exemple la liste *incList* pour les dépendances fonctionnelles [BHdL11] ou *keyTreeTemp* pour les clés [BCF+07]. A la fin, un test final vérifie que ces structures auxiliaires ne contiennent plus aucune violation (c'est-à-dire que chaque violation temporaire a bien été supprimée du fait d'une opération de mise-à-jour subséquente à celle qui l'avait suscitée).

Algorithme 2 - Validation incrémentale d'une contrainte d'intégrité C

Entrées :

(i) t : arbre XML
(ii) *UpdateList* : séquence de mises-à-jour
(iii) C : la contrainte d'intégrité
(iv) MEF_C : ensemble des machines à états finis correspondant à la contrainte C
(v) Aux : structure auxiliaire résultat de la dernière validation de t par rapport à C (le n-uplet de validation)

Sortie :

Si t reste valide alors *true* (de plus l'arbre t mis à jour remplace l'arbre donné et la structure Aux mise à jour remplace la structure Aux donnée) sinon $false$ (et t et Aux inchangés).

Variables locales :

(i) $CONF$: stocke les valeurs des attributs hérités
(ii) $SYNT$: stocke les valeurs des attributs synthétisés
(iii) *tempPb* : violations temporaires
(iv) *tempAux* : structure Aux temporaire

1. Initialize *tempPb*, *tempAux*, $CONF_\epsilon$ (using MEF_C)
2. **for each** *event v in t*
3. **switch** (v) **do**
4. **case** *open tag in position p*
5. Compute $CONF_p$ using MEF_C
6. **if** $\exists\ u = (p, \mathsf{insert}, t') \in UpdateList$
7. **then if** $\neg insert(C, t, p, t', MEF_C, tempAux, tempPb)$ **then** return false

25. Valeurs remontées depuis les feuilles.

8. Analyse the updates on *attributes* of this element at position p, if any
9. **if** $\neg\exists u' = (p'', op', t'') \in$ *UpdateList such that* $p \prec p''$ **then** $skipSubTree(t, p)$;
10. **case** *closing tag in position* p
11. Compute $SYNT_p$ using $CONF_p$
12. **if** $\exists\ u = (p, \mathsf{delete}) \in$ *UpdateList*
13. **then if** $\neg delete(C, t, p, SYNT_p, tempAux, tempPb)$ **then** return false
14. **if** $\exists\ u = (p, \mathsf{replace}, t') \in$ *UpdateList*
15. **then if** $\neg replace(C, t, p, t', SYNT_p, MEF_C, tempAux, tempPb)$ **then** return false
16. **if** $\exists\ u = (p.i, \mathsf{insert}, t') \in$ *UpdateList* where $p.i$ is a frontier position under p
17. **then if** $\neg insert(C, t, p.i, t', MEF_C, tempAux, tempPb)$ **then** return false
18. **case** *attribute* or *value* in position p
19. Compute $SYNT_p$ using $CONF_p$ and $value(p)$
20. **end for**
21. **if** *tempPb* indicates violation **then** return false
22. update the auxiliary structure *Aux* using *tempAux* and *tempPb*
23. **return** true □

Dans l'algorithme 2, à la rencontre d'une balise ouvrante les valeurs de $CONF_p$ sont calculées (il s'agit d'un très faible nombre de transitions dans un automate à états finis, c'est immédiat) et les éventuelles opérations d'insertion en cette position sont considérées, consistant ici à glisser le nœud courant et son sous-arbre vers la droite, sans qu'il soit nécessaire de descendre dans ce sous-arbre avant de réaliser les tests pour l'insertion (ligne 7). A ce stade il faut traiter les attributs de l'élément considéré. La ligne 9 permet d'éviter de traiter des sous-arbres entiers qui ne sont impliqués dans aucune mise-à-jour. Lorsqu'une balise fermante est atteinte en position p, les tests concernant les opérations de suppression et de remplacement sont effectués car à ce stade les attributs synthétisés nécessaires aux tests ont été instanciés dans $SYNT_p$. C'est également là que les insertions en frontière sous le nœud qui vient d'être considéré sont prises en compte.

Les fonctions *insert*, *delete* et *replace* réalisent les actions nécessaires pour vérifier la validité, qui sont différentes pour chaque sorte de contrainte (c'est pourquoi ces fonctions prennent la contrainte elle-même en paramètre). Les détails en ont été publiés dans [BCF+07] pour les clés et clés étrangères et dans [BFL12] pour les dépendances fonctionnelles. Le principe général en est le suivant : pour vérifier une insertion ou une suppression d'arbre il faut :
(A) Vérifier si l'arbre à ajouter est localement valide, donc calculer le n-uplet de validation pour chaque arbre à ajouter (et aussi chaque arbre en remplaçant un autre) et pour chaque arbre à supprimer également car ce n-uplet contient des valeurs utiles aux tests de validité à réaliser.
(B) Trouver la position du nœud contexte concerné par l'opération de mise-à-jour et analyser sa position par rapport à la position de la mise-à-jour. Parfois le n-uplet de validation calculé à l'étape (A) doit être complété par d'autres valeurs existantes dans le même contexte avant d'être testé.
(C) Trouver dans les structures auxiliaires l'information nécessaire pour décider si la mise-à-jour induit une violation de contrainte (qui peut être temporaire), ou pas.
(D) Conserver l'information sur les violations temporaires pour l'analyser à nouveau à

chaque nouvelle opération de mise-à-jour traitée, jusqu'à la fin de la transaction complète.

La complexité de ces opérations et du processus global de validation incrémentale pour les clés, clés étrangères et dépendances fonctionnelles a été analysée dans [BCF+07, Lim07, BFL12]. L'opération la plus coûteuse est toujours la recherche (linéaire) dans la structure auxiliaire qui rassemble les informations sur les violations temporaires (qui peut au pire atteindre la taille du document). Pour les dépendances fonctionnelles, plus complexes à vérifier que les clés, si on considère que le parcours de cette structure est en $O(N)$, N étant la taille du document à vérifier (c'est un pire cas jamais réalisé en pratique) alors pour n contraintes la complexité des vérifications dans cette structure auxiliaire est en $O(n.N)$. La complexité de la validation incrémentale est alors en $O(m(\mid t_1 \mid + n.N))$, où m est le nombre d'opérations de mise-à-jour et $\mid t_1 \mid$ est le temps moyen de validation locale d'un sous-arbre. Nos expérimentations rapportées dans [BFL12] confirment que la validation incrémentale de dépendances fonctionnelles est linéaire sur le nombre de contraintes en jeu et sur le nombre d'opérations de mise-à-jour : par exemple pour 100 mises-à-jour sur un document contenant 15 000 n-uplets pour 3 dépendances fonctionnelles différentes, le temps de validation est de 3 secondes (ce qui est mieux que celui de la validation initiale - 7 secondes -).

Un tel cadre générique pour la définition et surtout la validation, y compris incrémentale, des contraintes d'intégrité pour XML n'existe pas par ailleurs dans la littérature. Il représente une contribution importante à la recherche d'une définition expressive et d'une gestion efficace des données exprimées dans des formats XML, dans l'environnement en constante évolution qu'est le web.

1.5 Conclusion et perspectives

La recherche sur la gestion des données XML a été très riche dans les années 2000 et reste active comme en témoignent plusieurs publications chaque année dans les meilleures conférences fondamentales (ICDT, PODS, SIGMOD, VLDB) et dans de nombreuses autres plus appliquées. Ainsi par exemple pour les contraintes d'intégrité il y a tout récemment à SIGMOD 2013 un article sur la découverte de clés XML [ADN+13] et le meilleur article de la conférence DEXA 2012 [FHL+12] décrit une analyse qui permet de décider efficacement de *l'équivalence* entre plusieurs contraintes de clés, évitant ainsi un certain nombre de vérifications effectives par un système tel que le nôtre. Il en va de même pour la question abordée par la correction de document XML, qui a fait également l'objet de communication récentes comme par exemple [PRS12] à ICDT 2012.

Mes perspectives concernant les contraintes d'intégrité sont de deux ordres, d'une part continuer à étendre le cadre générique de définition et validation présenté et d'autre part le transposer ou l'appliquer dans un cadre différent de la seule validation. Dans la première direction il s'agit d'enrichir le pouvoir d'expression du langage de spécification des contraintes, pour distinguer par exemple des arborescences comme $r(a(b,c),a(b,d))$ et $(r(a(b,c,d))$. Une approche de test de validité de mises-à-jour complémentaire à la nôtre est présentée dans [GI10], fondée sur des vérifications *statiques* sur les définitions de la contrainte et de la mise-à-jour, toutes deux *motifs réguliers d'arbre*. Au-delà des conclu-

sions qui peuvent être déduites de cette analyse statique il reste une partie dynamique qui nécessite un accès aux données, qui pourrait alors reposer sur notre cadre pour être réalisée. A noter que dans [Hal07] les avantages des requêtes d'arbres simples (de type *tree pattern* [BFK03]) ont également été étudiées.

Mes perspectives dans la deuxième direction supposent un changement de point de vue, par exemple je pense que le cadre des requêtes arbres que [GI10] utilise pour exprimer des contraintes peut servir à décrire précisément des *préférences* concernant des documents XML recherchés ; de telles préférences peuvent alors être combinées à la requête comme les contraintes sont combinées à la requête représentant la vue XML dans [GI10]. Par ailleurs, j'envisage d'étudier dans quelle mesure notre contribution peut être utilisée dans le domaine connexe de la réparation de violation de contraintes par des mises-à-jour, abordé par exemple dans [TZ11], selon le même raisonnement qui nous avait amenées à étudier les corrections possibles des mises-à-jour en ce qui concerne la validité par rapport à un schéma et qui a abouti à l'algorithme général de correction présenté dans ce mémoire.

Mes perspectives concernant cet algorithme de correction sont liées à l'état de l'art approfondi que nous avons réalisé dans [ABS13]. Comme pour les contraintes d'intégrité, il montre d'une part que notre cadre peut être étendu et que d'autre part il peut être transposé ou appliqué à bien des situations. Pour les extensions, il s'agit d'étudier par exemple comment intégrer d'autres opérations d'édition-nœud qui agissent sur les niveaux de l'arbre (création ou suppression d'un niveau) ou sur des interversions entre nœuds frères ([Suz07, SFC08, SM11]). Pour les applications, je pense que cet algorithme peut être repris et adapté pour servir dans le contexte de l'interopérabilité des données (Data Exchange en anglais) [BBR11], activité qui consiste à transformer une donnée respectant une structure (schéma ou type ou classe) pour l'adapter à une autre structure cible.

Les deux contributions à la gestion des données XML que je viens de présenter sont complémentaires car traitant de la structure d'une part et d'autre part des données contenues dans un document XML. Elles partagent l'objectif de maîtrise des données et de leur évolution, par des contrôles automatiques définis à partir des *déclarations* spécifiées lors de la phase de conception, phase que j'aborderai dans le chapitre suivant.

Chapitre 2

Démarche de conception pour XML

Dans le premier chapitre, nous avons considéré le format XML en tant que niveau logique, d'un œil de gestionnaire de données, en considérant d'abord la structure des données et ensuite les contraintes d'intégrité sur les données. Classiquement, un modèle logique de données est un intermédiaire entre (i) le modèle conceptuel des données, suffisamment abstrait pour permettre aux utilisateurs de l'informatique (producteurs et consommateurs de données) une réflexion correcte et productive et (ii) le modèle physique d'implantation des données, axé sur l'efficacité des traitements machine. L'intérêt du niveau logique est de reposer sur une théorie mathématique qui permet de définir un certain nombre de *vérifications automatiques* de correction et de cohérence des données. C'est le cas pour XML comme pour le modèle relationnel.

Mes contributions rappelées dans le chapitre précédent abordent donc XML sous cet angle, mais ma première approche de XML, à la fin des années 90, était motivée par la *représentation de données*, en particulier pour le traitement automatique des langues naturelles [1]. Par la suite, pour les projets menés par Denis Maurel, j'ai proposé des schémas XML pour représenter les noms propres et leurs relations, dans la chaîne de traitements Unitex [2] et surtout au sein d'une ressource lexicale multilingue [BTM05, BM08, MB13].

Dans ce chapitre je considère la question de la *conception de bases de données XML* et l'éventail de réponses qui lui correspondent dès lors qu'il s'agit d'XML. Je commence en section 1 par aborder les démarches classiques de conception transposées au contexte d'XML. Puis je présente le cadre d'application, l'ontologie des noms propres et de leurs relations, matérialisée dans la base de données Prolexbase [Tra06, Mau08]. Je décris ensuite la transcription de ses concepts dans le modèle conceptuel LMF [ISO08], conçu et adopté pour favoriser l'interopérabilité. Enfin je présente la conception du format XML de Prolmf, au regard des démarches de conception classiques et des propositions élaborées par la communauté du traitement automatique des langues.

1. Les données concernées étaient les "prédicats et classes d'objets" tels que définis par Gaston Gross [PMC98], pour les utiliser pour une interrogation de bases de données en langue naturelle [BM99a, BM99b, BKM99b, BLM01].
2. `www-igm.univ-mlv.fr/~ unitex/`

2.1 Démarches de conception et XML

Nous considérons ici les deux démarches de conception classiquement connues pour mener à un bon modèle logique de données, c'est à dire à un modèle normalisé pour limiter les risques de perte d'intégrité des données. Ces deux démarches de conception sont : (i) partir d'une modélisation des données au niveau conceptuel (schémas Entités-Associations ou UML) qui respecte certaines contraintes puis le traduire vers le niveau logique, démarche sans doute la plus pratiquée, et (ii) une conception directe au niveau logique ce qui, dans l'univers des bases de données relationnelles, consiste à définir un ensemble de relations et un ensemble de dépendances de données sur ces relations. On peut alors s'appuyer sur la théorie des dépendances fonctionnelles pour parvenir à une des formes normales connues pour éviter les redondances et anomalies de mises-à-jour [Dat03].

2.1.1 Passage d'un modèle conceptuel à un schéma XML

Contrairement à la démarche standardisée de passage d'un schéma conceptuel à un schéma relationnel, la dérivation d'un schéma XML à partir d'un schéma conceptuel n'est pas quasiment automatique. Ceci est dû au fait qu'il y a beaucoup de représentations XML possibles de certaines parties des schémas conceptuels, en particulier *les relations entre les classes* UML (ou les associations entre entités). Nous avons rencontré cette difficulté lorsque nous avons conçu une représentation XML de Prolexbase indépendamment de LMF [BTM05]. Nous avons pu alors constater que le choix de la meilleure représentation est fonction de l'usage pour lequel est conçu le schéma XML.

La question n'a pas laissé les chercheurs indifférents, car commencer une conception par une modélisation conceptuelle est une démarche largement adoptée et dont les avantages sont évidents. Malheureusement, dans [DLP+07] les auteurs recensent jusqu'à 22 catégories de transformations d'un modèle de classes UML vers un schéma XML (en XML Schema), plusieurs propositions assez semblables par des auteurs différents ayant été regroupées dans une même catégorie. Ces propositions de transformations ont été publiées entre 1999 et 2007 (année de rédaction de l'article cité). Ce tour d'horizon comparatif montre la grande diversité des solutions possibles d'une part, et d'autre part que la proposition qui couvre le plus d'éléments de modélisation UML est celle décrite dans [KK03], issue d'une thèse consacrée à ce sujet. La table 2.1 est un extrait de celles de [DLP+07]. Elle illustre bien la multiplicité combinatoire des solutions possibles (on n'a gardé ici que les éléments les plus courants d'un diagramme de classes UML).

Notons que lorsqu'on s'intéresse à la question du passage d'un modèle UML à XML, on rencontre inévitablement XMI (XML Metadata Interchange [OMG02]), un standard de l'OMG [3]. Il s'agit d'un ensemble de règles pour décrire dans un format XML les éléments d'un modèle exprimé dans un formalisme de la famille des MOF (Meta Object Facilities), dont fait partie UML. Ce standard est donc définit pour *l'échange de modèles via XML*. La représentation XMI n'est donc pas spécialement le format dans lequel seront structurées *les données décrites par le modèle UML* mais plutôt la représentation en XML du *modèle UML*. Néanmoins les règles décrites permettent de produire un schéma XML des données

3. Object Management Group, à l'origine d'UML, MDA, CORBA et autres standards liés à la conception orientée objet. Pour XMI voir : `www.omg.org/spec/XMI/`

UML	**Schéma en XML Schema**
classe	type complexe ou élément (global) ou les 2
attribut de classe	élément dans la plupart des propositions, parfois élément ou attribut selon la nature de l'attribut UML
association	il y a 4 options différentes :
	(1) un élément (ou type) est imbriqué dans l'autre
	(2) des contraintes de clés et de références (key/ref) sont ajoutées dans les 2 éléments (ou types) associés
	(3) un élément est créé pour l'association, contenant des références vers les 2 éléments associés
	(4) des références utilisant les standards XLink et XPointer sont ajoutées
généralisation	Il y a 4 possibilités, la 2ème étant la plus souvent choisie :
	(1) toutes les classes sont transformées en type complexe et éléments de ces types ; la hiérarchie est transformée en imbrication des éléments
	(2) la classe racine est transformée en type complexe et élément de ce type ; les autres classes en types complexes dérivés par extension du type de la super classe
	(3) la classe racine est transformée en type complexe et élément de ce type ; les sous classes sont traduites en types dérivés par restriction du type de la super classe
	(4) des éléments sont déclarés pour chaque classe et ils sont organisés dans un groupe de substitution
composition	l'élément composant est imbriqué dans l'autre
agrégation	Il y a 3 possibilités :
	(1) élément imbriqué dans la classe agrégeante pour référencer l'élément (global) représentant la classe aggrégée
	(2) élément indépendant et attributs pour faire les liens
	(3) attributs
multiplicité d'un attribut	attributs minOccurs / maxOccurs de l'élément représentant l'attribut
multiplicité d'une association	attributs minOccurs / maxOccurs là où est représentée l'association
restrictions sur la valeur d'un attribut	restrictions sur le type de l'élément
type non prédéfini	type complexe

Table 2.1 – correspondances entre éléments de diagrammes UML et items de schémas XML

dans la mesure où les concepts de UML sont décrits en UML avant d'être représentés en XML, les règles correspondent donc à des choix de représentation parmi ceux listés dans la table 2.1. Par exemple, toute classe du modèle est représentée par (i) un élément (global) ayant le nom de la classe et (ii) un type complexe nommé du nom de la classe (3ème option de la 1ère ligne du tableau 2.1). Les propriétés de la classe peuvent être des éléments XML ou des attributs XML. Une association est représentée par un élément XML contenant des références vers les éléments XML associés (cf. option (3) pour les associations dans le tableau 2.1). Etc.

2.1.2 Théorie de la normalisation pour XML

Comment atteindre pour XML les objectifs de la normalisation en relationnel : éliminer les ambiguïtés, minimiser les redondances, préserver la cohérence des données et permettre une maintenance rationnelle ? C'est le problème auquel se sont attelés toute une série de travaux, initiés au début des années 2000 par les propositions de L. Libkin et M. Arenas sur la forme XNF, une forme normale de documents XML [AL02, AL04, Are06, Lib07, Are09], renforcés par celles de M. W. Vincent et J. Liu qui en ont proposé une autre (4NF), et poursuivent des travaux sur XNF [VLL04, VL05, VLM07, VLM12] [4]. D'un autre côté, C. Yu et H.V. Jagadish l'ont prise sous un angle différent, en proposant de détecter automatiquement les dépendances fonctionnelles par data mining sur les documents XML eux-mêmes [YJ08], ce que viennent compléter les propositions toutes récentes d'Arenas et ses collègues pour la découverte de clés [ADN+13].

Marcelo Arenas et Leonid Libkin motivent ainsi leurs travaux initiaux : poser les principes d'une bonne conception XML, c'est à dire à la fois d'une conception du schéma approprié aux objectifs, schéma qu'ils considèrent décrit par une DTD, et d'un ensemble de dépendances de données. Pour cela, ils définissent une forme normale pour les documents XML de la façon suivante :

Définition 2.1.1 - *XNF [AL02]* : Etant donnés une DTD D et un ensemble Σ de dépendances fonctionnelles définies en fonction de D [5], (D, Σ) est en forme normale XNF si et seulement si pour toute XDF non triviale $X \rightarrow p.@a \in (D, \Sigma)^+$, on a aussi $X \rightarrow p \in (D, \Sigma)^+$, où X représente un ensemble d'éléments (ou attributs).

L'intuition est la suivante : si σ, la dépendance indiquant que X détermine la valeur de l'attribut a de p, est impliquée par (D, Σ) alors si T est un arbre XML valide par rapport à la DTD D et s'il satisfait σ, alors dans T, quels que soient les ensembles de valeurs des éléments spécifiés dans X, on ne peut trouver qu'une seule valeur de l'attribut a de p. Donc, pour éviter de stocker cette valeur de façon redondante pour tous les ensembles de valeurs des éléments spécifiés dans X, il faut la stocker une seule fois, de sorte que $X \rightarrow p$ soit impliquée par (D, Σ).

Concrètement, lorsqu'une valeur est répétée dans beaucoup de sous éléments d'un élément donné, les auteurs proposent de remonter les attributs répétés jusqu'au parent qui

4. La question a également été abordée par K. D. Scheeves [Sch05] et J. Wang et R. Topor [WT05].

5. Cette définition revient à celle présentée au chapitre 1, sauf que ces auteurs utilisent la DTD pour spécifier un équivalent de nos motifs.

regroupe toutes les répétitions. Par exemple, soit la DTD D suivante, représentant une conférence :

```
<!DOCTYPE db [
  <!ELEMENT conference (edition*)>
  <!ELEMENT edition (article*)>
  <!ELEMENT article (auteur+, titre)>
    <!ATTLIST article annee CDATA #REQUIRED>
... ]>
```

On voit que dans D l'année est répétée dans les articles d'une même édition : en considérant la dépendance fonctionnelle
$DF_1 = conference/edition \rightarrow conference/edition/article@annee$
indiquant que tous les articles d'une édition sont de la même année, le couple $(D, \{DF_1\})$ n'est pas en XNF car $conference/edition \rightarrow conference/edition/article$ n'est pas impliquée par $(D, \{DF_1\})$.

Par contre, dans la DTD D' identique à D sauf que l'attribut *annee* est au niveau de l'édition et non de l'article, la dépendance fonctionnelle s'écrit
$DF'_1 = conference/edition \rightarrow conference/edition@annee$
et donc $(D', \{DF'_1\})$ est en XNF.

De même, certains raisonnements connus en relationnel sont transposés à XML comme dans l'exemple suivant (issu de [Lib07]) :

```
<!DOCTYPE r [
  <!ELEMENT r (tuple*)>
  <!ELEMENT tuple (EMPTY)>
    <!ATTLIST tuple
              A CDATA #REQUIRED
              B CDATA #REQUIRED
              C CDATA #REQUIRED>
]>
```

Avec les dépendances fonctionnelles $DF_2 = \{r.tuple.@A, r.tuple.@B\} \rightarrow r.tuple.@C$ et $DF_3 = r.tuple.@C \rightarrow r.tuple.@A$, ce modèle n'est pas en forme normale pour les mêmes raisons qu'en relationnel, et sa transformation vers une forme normale suivra la démarche connue en relationnel, à savoir une décomposition, par exemple de la manière suivante :

```
<!DOCTYPE r [
  <!ELEMENT r (tuple*, rNew)>
  <!ELEMENT tuple (EMPTY)>
    <!ATTLIST tuple
              A CDATA #REQUIRED
              C CDATA #REQUIRED>
  <!ELEMENT rNew (tupleNew*)>
  <!ELEMENT tupleNew (EMPTY)>
    <!ATTLIST tupleNew
              B CDATA #REQUIRED
              C CDATA #REQUIRED>
]>
```

Dans les mêmes références, les auteurs proposent d'utiliser la théorie de l'information de

Shannon pour évaluer la quantité de redondance dans un modèle et ils montrent que cette proposition mesure effectivement la qualité du modèle, indépendamment de tout langage de requête et de mise-à-jour, ceci tant dans le domaine relationnel que dans le domaine XML. L'idée est la suivante : pour mesurer la quantité d'information dans la position p par rapport à un ensemble quelconque de positions P dans le document T instance du modèle, on suppose qu'on a perdu la valeur de la position p et qu'on a un ensemble de k valeurs possibles $v_1, ..., v_k$ parmi lesquelles il faut choisir la valeur de p. Il faut assigner à chaque v_i une probabilité π_i d'être la valeur de p, étant donnée l'information fournie par les positions dans P. Pour calculer π_i, il faut considérer les dépendances fonctionnelles. Inversement, en supposant que la valeur de p soit v_i et que les valeurs des positions dans P aient été perdues, les pourcentages des nombres d'affectations de valeurs des positions dans P (à partir du même ensemble $v_1, ..., v_k$) qui amènent l'instance résultante à satisfaire toutes les DF sont également les π_i (que l'on aura dû normaliser pour ramener leur somme à 1). Ainsi, par un raisonnement sur les dépendances fonctionnelles et sur la structure, on peut déterminer la qualité d'un modèle relationnel ou XML.

Arenas et Libkin ont montré que la forme XNF généralise la forme BCNF définie dans le cadre relationnel, la complexité du problème consistant à tester si un modèle XML (schéma et dépendances de données) est en XNF restant pour eux une question ouverte.

Millist W. Vincent et Jixue Liu travaillent également sur la définition de dépendances fonctionnelles pour XML et leur utilisation dans la définition de formes normales, depuis une dizaine d'années [VLL04, VL05, VLM07, VLM12]. Contrairement à Arenas et Libkin ils ne font pas reposer leurs définitions de dépendances fonctionnelles sur celle d'une DTD mais proposent des définitions qui reviennent à celles présentées au chapitre précédent. Ils ont en particulier traité très soigneusement le *problème de l'implication* des dépendances fonctionnelles, pour XML mais également, en retour, pour le modèle relationnel.

Dans leur dernier article [VLM12], ils reviennent sur le lien entre dépendances fonctionnelles et DTD et montrent que certaines DTD peuvent se ramener à un ensemble de leurs dépendances fonctionnelles. Dans le même article, ils développent et prouvent la correction et la complétude d'un système d'implication logique (système d'axiomes), pour une définition de dépendance fonctionnelle dite "du nœud le plus proche", contrainte qui généralise la notion de dépendances fonctionnelles pour une base de données relationnelle. Ils proposent à partir de cet ensemble d'axiomes un algorithme d'implication de dépendances fonctionnelles correct et complet, qui calcule la fermeture transitive d'un ensemble de dépendances fonctionnelles en temps quadratique sur le nombre et la taille des dépendances (nombre et taille des chemins).

C. Yu et H. V. Jagadish adoptent une démarche différente dans [YJ08], en proposant de détecter les redondances directement dans le document XML, pour le transformer ensuite dans une forme normale. Ils définissent la notion de redondance dans un document XML en s'appuyant sur une définition de dépendance fonctionnelle qui fait appel à la notion de n-uplet d'arbre généralisé appelé GTT. Ils présentent alors leur forme normale de schémas XML appelée GTT-XNF de la façon suivante : un document T contient une redondance si et seulement si il satisfait une dépendance fonctionnelle non triviale (C_p, LHS, RHS) mais ne satisfait pas la clé (C_p, LHS). Intuitivement, si (C_p, LHS) n'est pas une clé pour T alors il existe 2 n-uplets distincts dans C_p qui ont le même LHS. Comme T satisfait

par contre la dépendance fonctionnelle (C_p, LHS, RHS), les chemins RHS des 2 n-uplets doivent être égaux en valeur. Cependant ces chemins sont distincts (car ils partent de 2 nœuds pivots distincts), leurs valeurs sont donc redondantes.

A partir de ces notions, les auteurs présentent un algorithme $DiscoverXFD$ pour découvrir des redondances dans un document XML, redondances exprimées sous la forme de dépendances fonctionnelles. $DiscoverXFD$ repose sur une transformation du document XML en relationnel (plusieurs transformations sont discutées), permettant d'exploiter ensuite des algorithmes connus de découverte de dépendances dans une base de données relationnelle. Puis ils proposent un autre algorithme pour transformer le document XML en forme GTT-XNF.

La complexité de $DiscoverXFD$ pour une relation R est en $O(R_n R_k 2^{R_k})$ où R_n est le nombre de n-uplets dans R et R_k est le nombre d'attributs dans R. Celle de la transformation n'est pas évaluée. Les expérimentations menées avec $DiscoverXFD$ montrent qu'il est praticable (mais long) sur des documents XML volumineux. L'algorithme de transformation vers une forme normale n'a pas été implanté. Toutefois ses principes peuvent au moins guider une démarche de conception : pour éliminer une redondance détectée du fait que le document T satisfait une dépendance fonctionnelle non triviale (C_p, LHS, RHS) mais ne satisfait pas la clé (C_p, LHS), ils déplacent les éléments correspondant à RHS vers une zone du document où ils n'auront plus à être dupliqués, et ils créent une contrainte d'inclusion pour garder le lien de ces éléments avec leur contexte initial.

C'est ce qui est réalisé pour les auteurs et le titre des livres dans l'exemple suivant, extrait de [YJ08] [6] :

```
<!DOCTYPE warehouse [
   <!ELEMENT warehouse (state*)>
   <!ELEMENT state (name, store*)>
   <!ELEMENT store (contact, book*)>
   <!ELEMENT contact (name, address)>
   <!ELEMENT book (ISBN, author+, title, price)>
... ]>
```

Les XDF $F1$, $F2$ et $F3$ suivantes sont écrites dans leur syntaxe de XDF et indiquent, respectivement, que (1) dans le contexte d'un élément *book* la valeur de l'ISBN détermine celle du titre, (2) dans le contexte d'un élément *book* la combinaison des valeurs (nom du magasin, ISBN) détermine la valeur du prix, et (3) dans le contexte d'un élément *book* la valeur de l'ISBN détermine celles des auteurs.

F1 = $\{./ISBN\} \rightarrow ./title$ w.r.t. C_{book} ou $(C_{book}, \{./ISBN\}, ./title)$

F2 = $\{../contact/name, ./ISBN\} \rightarrow ./price$ w.r.t. C_{book}

F3 = $\{./ISBN\} \rightarrow ./author$ w.r.t. C_{book}

Alors en supposant que la clé $C_{store}, \{../name, ./contact/name\}$ [7] soit vérifiée, l'algorithme de transformation destiné à éliminer les trois XDF précédentes produit le schéma suivant, dans lequel on retrouve le principe de ne pas répéter des informations dans de multiples sous-éléments mais de les factoriser à un niveau, puis d'y faire référence (ici via

6. Ces auteurs utilisent une notation de schéma particulière mais nous reproduisons ici leur schéma sous forme de DTD pour des raisons d'homogénéité.

7. Un magasin est identifié par le nom de l'état où il se trouve et le nom de son sous-élément contact.

l'ISBN) :

```
<!DOCTYPE warehouse [
    <!ELEMENT warehouse (state*,bookNew*)>
    <!ELEMENT state (name, store*)>
    <!ELEMENT store (contact, book*)>
    <!ELEMENT contact (name, address)>
    <!ELEMENT book (ISBN, price)>
    <!ELEMENT bookNew (ISBN, author+, title)>
... ]>
```

Comme déjà vu dans le chapitre 1, la recherche sur les dépendances fonctionnelles en XML et leur application à la recherche de formes normales est active. Pourtant, de la même façon qu'il n'est pas immédiat de dériver un schéma XML à partir d'un schéma conceptuel, il n'y a pas non plus de méthode de transformation complète qui amène à un schéma XML qui éviterait toute redondance. Néanmoins, tout comme pour la dérivation "conceptuel vers logique" il existe des propositions sur lesquelles s'appuyer, même de façon informelle, pour parvenir à un meilleur schéma.

Une conclusion de ce rapide tour d'horizon est que la question de la conception de schéma XML a été l'objet de recherches actives depuis la publication d'XML. Notons que ce tour d'horizon est limité par son objectif, construire un schéma XML à partir d'une modélisation UML, mais qu'il y a encore beaucoup d'autres travaux sur des démarches menant à la production d'un schéma : nous avons évoqué dans le chapitre 1 les propositions qui s'appuient sur des adaptations de schémas existants, mais il y a encore celles centrées sur des inférences de schéma à partir de documents (entre autres [BNV07]), celles sur les "résumés de données", ou encore celles concernant la mise en correspondance (mapping) et l'intégration de schémas XML ([MPB08, BBR11]).

Par ailleurs, dès la fin des années 90 la transformation de données relationnelles vers des documents XML a été étudiée, par exemple dans [FTS00]. Les schémas cibles et les mappings étaient d'abord définis par des experts, la génération d'un schéma XML qui garantisse la conservation des propriétés d'un modèle relationnel (ensemble de relations et ensemble de dépendances de données sur ces relations) respectant une des formes normales connues se révélant être un challenge qui n'a pas non plus de solution immédiate mais, encore une fois, de multiples solutions possibles. Deux algorithmes de génération automatique, produisant des schémas relativement limités, sont présentés dans [LMCC02], avec des résultats expérimentaux qui valident les idées mais restent prohibitifs en termes de temps de calculs. Ainsi la solution qui consisterait (i) à concevoir un bon schéma relationnel, par l'une ou l'autre des deux approches classiques performantes dans le cadre relationnel, puis (ii) à générer ensuite un schéma XML à partir de ce (bon) schéma relationnel ne serait pas plus évidente à mettre en oeuvre que les autres approches.

Pour conclure cette section sur la conception de schémas XML d'un point de vue général, rappelons que la question "comment construire de bons documents (ou schémas) XML" fut active dans la première moitié des années 2000 dans les forums de discussion [8] et donna lieu à des initiatives de regroupements de recommandations telles que XML Design

8. Voir par exemple `http://msdn.microsoft.com/library/aa468564`.

Patterns [9] et surtout XML Schemas : Best Practices [10]. Par ailleurs, les éditeurs d'environnement de travail sur des documents XML comme Altova XMLSpy [11], Stylus Studio [12], ou oXigen [13], reprennent ces recommandations dans leurs outils d'assistance et de documentation. Des recommandations particulièrement bien étayées sur des exemples concrets de réalisations à grande échelle se trouvent également dans [Wal12], globalement toutes ces recommandations ont pour objectifs :

- (1) de rendre les documents et leurs schémas plus simples à comprendre, à modifier et à réutiliser,
- (2) de garantir via le schéma une certaine qualité des données (typage précis, structure cohérente, intégrité des données),
- (3) de rendre les documents (et les schémas) flexibles et extensibles.

Comme on l'aura déjà compris, dans le cadre XML toute recommandation générale est à reconsidérer en pratique en fonction de multiples paramètres : (1) les contextes et scénarios d'utilisation (application orientée données ou non), (2) la cible principale (humains ou machines), (3) le formalisme utilisé pour décrire le schéma (DTD, XML Schema ou Relax-NG), (4) les technologies utilisées : une structure conviendra mieux en termes de facilité de programmation et maintenance, ou d'efficacité d'exécution, selon qu'on utilisera XSLT ou XQuery, SAX ou DOM, etc. La plupart des recommandations portent sur des aspects purement syntaxiques. En ce qui concerne les recommandations *méthodologiques*, celle qui revient le plus souvent dans l'ensemble des ressources citées est sans conteste celle d'adapter un schéma déjà éprouvé dans un contexte similaire plutôt que de développer un nouveau schéma à partir de rien.

Ce principe a fait l'objet de gros projets de constitution de cadres généraux de développement XML (schéma génériques et outils génériques), comme par exemple le projet américain "développement d'un modèle du cycle de vie" du NIST (US/DoC/NIST/MEL/MSID Manufacturing Systems Integration Division) [14]. On trouve désormais aisément des modèles de schéma pour représenter des informations courantes et il en existe également pour quantité de domaines particuliers, parfois extrèmement sophistiqués, dont on peut trouver une liste sur Wikipedia [15] : la TEI (Text Encoding Initiative) en fait partie.

2.2 Prolexbase

Notre objectif dans ce chapitre est la définition d'un schéma XML pour représenter une ressource lexicale conçue dans le cadre du projet Prolex. Le projet Prolex est une plateforme technologique pour le traitement automatique des noms propres, dont la première étape a été la constitution d'un dictionnaire multilingue des noms propres et de leurs relations, implanté sous la forme d'une base de données relationnelle, Prolexbase [Mau08].

9. http://www.xmlpatterns.com/
10. http://www.xfront.com/BestPracticesHomepage.html
11. http://www.altova.com/
12. http://www.stylusstudio.com/
13. http://www.oxygenxml.com
14. http://www.mel.nist.gov/msid/XML-testbed/index.html
15. http://en.wikipedia.org/wiki/List-of-XML-schemas

Ce dictionnaire est développé en partenariat avec la Faculté de Mathématiques de Belgrade pour le serbe [VKM07, VKM09] et l'IPIPAN [16] de Varsovie pour l'anglais et le polonais [SMB13]. Dans sa version actuelle il contient plusieurs centaines de milliers d'entrées [MB13, SMB13].

Une telle ressource est utile pour la plupart des applications en traitement automatique des langues naturelles, puisque les connaissances sur les noms propres permettent de situer le texte dans un contexte précis. Prolexbase représente bien plus qu'une simple liste de noms propres, elle comprend des descriptions linguistiques (morphologie, syntaxe) et sémantiques (synonymie, méronymie et relations sémantiques) précieuses pour la traduction automatique, la recherche d'information, etc. De plus, le traitement de l'aspect multilingue en est un point particulièrement intéressant (voir [TGM04, AGM+06, MVK07]).

2.2.1 Modèle ontologique

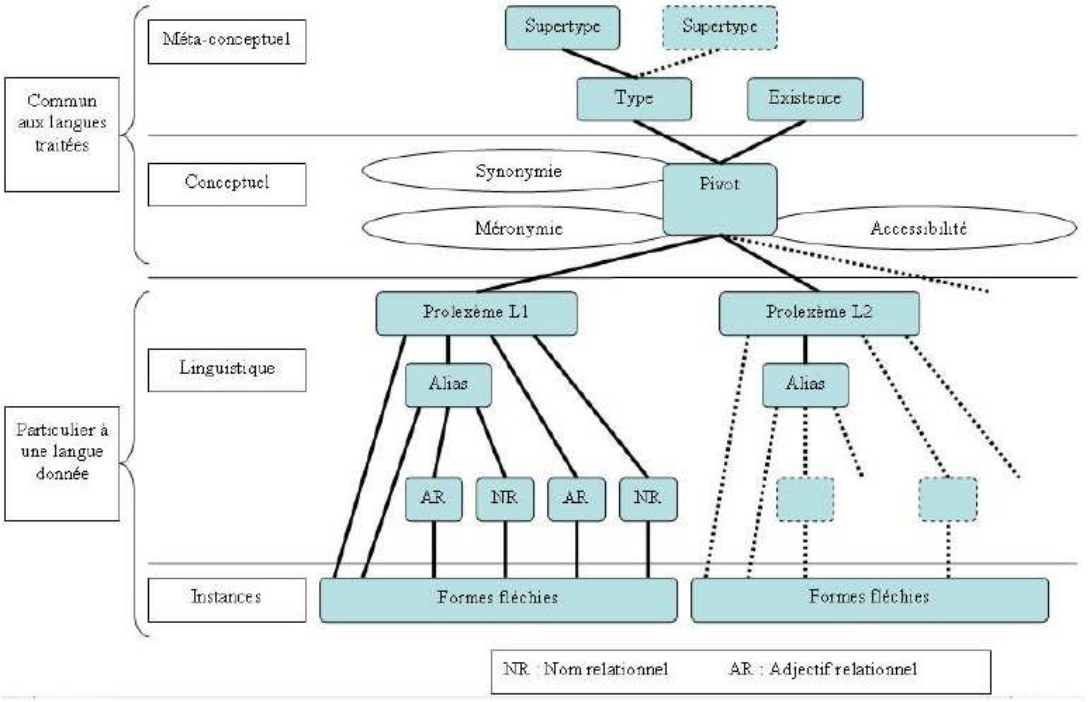


FIGURE 2.1 – Présentation abstraite de Prolexbase (ontologie [Tra06])

La figure 2.1 donne les principes d'organisation des informations dans Prolexbase, que nous rappelons brièvement maintenant (cf. [Tra06, Mau08, VKM09]). Au niveau d'une langue, Prolexbase contient des lemmes et des formes fléchies de noms propres, mais aussi d'alias et de dérivés de nom propre [17]. Par exemple, *Onu*, *Nations unies* et *Organisation des Nations unies* sont des alias d'un même nom propre tandis que *Parisien* et *Parigot*

16. Institute of Computer Science of the Polish Academy of Sciences.
17. Le *lien morphosémantique* dans WordNet [FM03]

sont des noms dérivés dont le sens se déduit régulièrement du nom propre *Paris*. Les alias peuvent être des synonymes dépendants de la langue (formes dialectales), des variantes orthographiques (majuscule ou pas, diacritique), des acronymes, des abréviations... Ils peuvent avoir des dérivés. Certaines langues ont une morphologie dérivationnelle riche pour les noms propres. Par exemple, il existe en serbe un adjectif possessif construit sur chaque nom à trait humain, y compris sur les noms de personne ou sur les noms d'habitant : *C'est la voiture d'un Parisien* peut se traduire par *To je Parižaninov auto*, où *Parižaninov* est un adjectif possessif [MVKK07].

L'originalité de Prolexbase est de rassembler l'ensemble {nom propre, alias, dérivés} sous une seule entrée linguistique appelée « prolexème », avec l'idée que, dans un contexte multilingue, la traduction d'un des mots de cet ensemble peut nécessiter l'utilisation d'un autre mot du prolexème de la langue cible. Si *Parisien* a une traduction en anglais, *Parisian*, il n'en va pas de même pour *Tourangeau* qui se traduira par une glose où sera présent le nom propre *Tours*, par exemple *inhabitant of the city of Tours in France*.

Chaque élément du prolexème a une catégorie, par exemple nom propre (N), alias (A), nom relationnel (NR), nom relationnel diastratique [18] (NRD), adjectif relationnel (AR), etc. En général le nom propre est choisi pour désigner l'ensemble : quand on parle du prolexème français *Paris* cela représente l'ensemble de lemmes Prolexème-fra$_{Paris}$={Paris.N, Parisien.NR, Parigot.NRD, parisien.AR, parigot.ARD}.

Par ailleurs, chaque lemme est associé à une règle de flexion, ce qui permet la génération de formes fléchies associées à un prolexème : Instances-fra$_{Paris}$={Paris.N:ms:fs, Parisien.N:ms, Parisienne.N:fs, Parisiens.N:mp, Parisiennes.N:fp, Parigot.N:ms, Parigote.N:fs, Parigots.N:mp, Parigotes.N:fp, parisien.A:ms, parisienne.A:fs, parisiens.A:mp, parisiennes.A:fp, parigot.A:ms, parigote.A:fs, parigots.A:mp, parigotes.A:fp}.

Pour le multilinguisme, chaque prolexème d'une langue est relié à *un et un seul pivot interlingue* qui est un identifiant unique. C'est par ce pivot que passe la traduction d'une langue à l'autre. A noter que le pivot correspond à un « point de vue sur le référent », aussi *Paris* et *Ville lumière* ont dans Prolexbase deux pivots différents. Par exemple, le pivot correspondant à Paris est 38 558 et c'est le pivot de chacun des prolexèmes Prolexème-fra$_{Paris}$, Prolexème-eng$_{Paris}$={Paris.N, Parisian.NR, Parisian.AR}, Prolexème-srp$_{Pariz}$={Pariz.N, Parižanin.NRM, Parižanka.NRF, Parižaninov.APM, Parižankin.APF, parižanski.AR...}. Pour passer d'une langue à une autre, suivant les mots et les catégories, on « traduit » ou on « glose » : le nom français *Paris* donnera *Paris* en anglais et *Pariz* en serbe, mais l'adjectif possessif serbe *Parižaninov* donnera *d'un Parisien* en français.

Les points de vue sur un référent représentés par les pivots sont liés les uns aux autres par deux relations paradigmatiques (synonymie [19] et méronymie) et par une relation syntagmatique (accessibilité). Par exemple le pivot 38 558 (*Paris*) est en relation de synonymie (diaphasique) avec le pivot de *Ville lumière*, en relation de méronymie avec le pivot de *Île-de-France* et en relation d'accessibilité (repérage « capitale ») avec le pivot de *France*. Ces trois relations sont complétées par des relations d'hyperonymie, soit vers une typologie, soit vers un paradigme d'existence. La figure 2.2 reprend les niveaux de description sur

18. d'un autre registre, familier pour *Parigot*.
19. Diachronique (entre *Zaïre* et *République Démocratique du Congo*) , diastratique (entre *Jean Paul II* and *Karol Józef Wojtyła*), et diaphasique (cf. exemple dans la suite du texte).

l'exemple du pivot 38 558 (*Paris*).

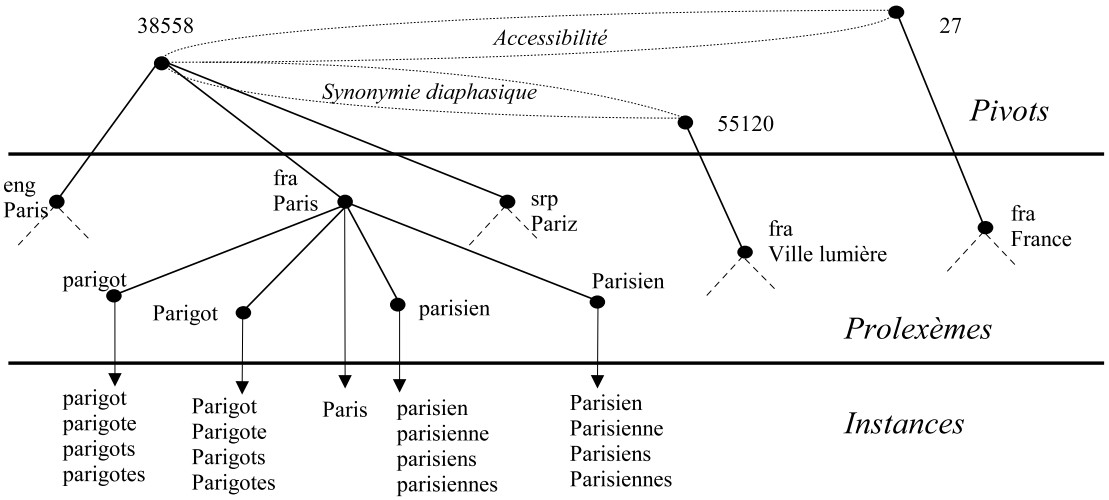


FIGURE 2.2 – L'entrée *Paris* dans Prolexbase

Deux relations syntagmatiques existent au niveau linguistique (propre à une langue) : l'expansion classifiante (*La ville de Paris*) et l'expansion d'accessibilité (*Paris est la capitale de la France*). Cette dernière est liée à la relation d'accessibilité, indépendante des langues. Enfin, des collocations, déterminants (*la* France) ou prépositions locatives (*en* France) peuvent préciser le contexte d'apparition d'un nom propre dans une langue donnée.

2.2.2 Modèles conceptuel, logique et physique

Prolexbase est une base de données implantée avec le système de gestion de bases de données MySql, selon la démarche de conception classique illustrée par la figure 2.3. Les descriptions de l'ontologie (en particulier la hiérarchie des types et les sortes d'existence) et les 3 niveaux de modèles (conceptuel, logique (relationnel) et physique) ont été développés dans le cadre de la thèse de Mickaël Tran ([Tra06]), puis plusieurs projets ont financé les séquences de saisies manuelles nécessaires au début. L'alimentation de cette base est en effet manuelle ou semi-automatique avec supervision [SMB13], ce qui, avec les règles de conception mises en oeuvre, garantit la qualité de ses données. Le modèle relationnel a évolué un peu depuis 2006 pour s'adapter aux applications développées autour de Prolexbase, son état actuel est décrit dans [SMS11].

Prolexbase est donc implémentée et maintenue sous la forme d'une base de données relationnelle (sous MySql). C'est sous cette forme qu'elle est régulièrement enrichie, son interface d'interrogation permet des exports prédéfinis et d'autres peuvent être faits pour telle ou telle application, à la demande. Le schéma relationnel actuel comprend 45 relations et une centaine de contraintes d'intégrité. Fin 2012 elle contient de l'ordre de 67 000 pivots et presque autant de relations, qui décrivent la sémantique de plus de 142 000 instances (noms, alias et dérivés) en français, 18 000 en anglais et 165 000 en polonais [SMB13].

Comme illustré figure 2.4, dès la conception il a été prévu d'offrir une possibilité d'export

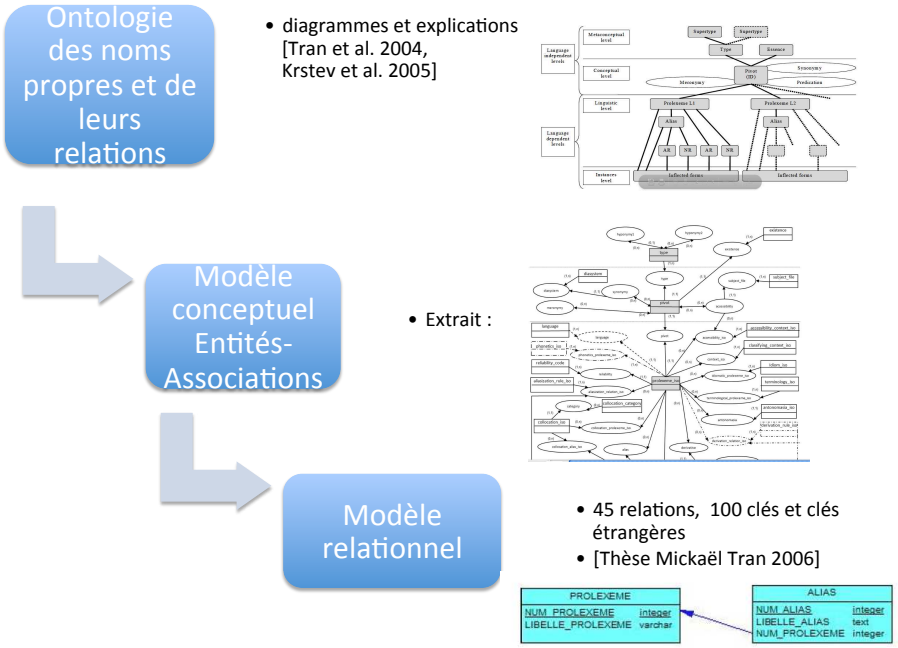


Figure 2.3 – La conception de Prolexbase (thèse de Mickaël Tran, 2006)

en format XML du contenu de la base, comme format standard d'échange de données. J'ai commencé par mettre au point un premier schéma XML pour représenter le contenu de la ressource, en partant de la description ontologique et du modèle conceptuel, mais aussi du modèle relationnel, en suivant les propositions de l'époque en matière (1) de passage vers le niveau logique XML et (2) d'utilisation des contraintes d'intégrité [BTM05]. Le schéma proposé alors transcrivait l'intégralité des détails prévus dans la thèse de Mickaël, et reflétait directement la description ontologique (figure 2.1) : une première partie du document devait contenir les informations des niveaux "méta-conceptuel" et "conceptuel", tandis qu'une deuxième partie devait accueillir les données, propres à chaque langage, des niveaux "linguistique" et "instances". Dans un langage donné, les instances (entrées lexicales) étaient rassemblées sous le prolexème (point de vue sur un référent, en d'autres termes un sens). En cela, ce premier schéma XML se rapprochait du standard TMF pour la représentation des ressources terminologiques, ISO 16642 :2003 [ISO03].

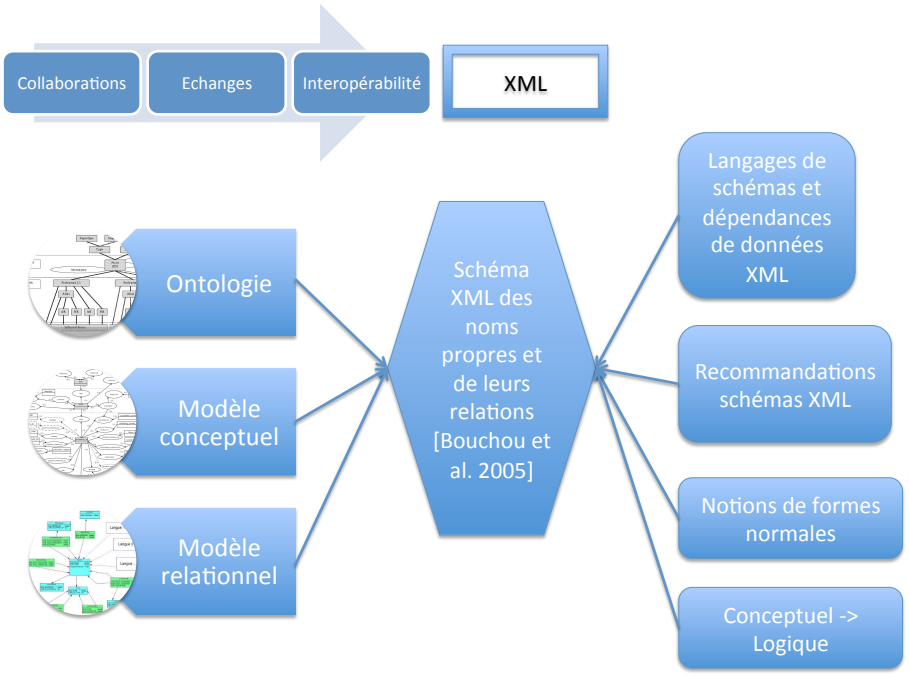


FIGURE 2.4 – Vers une représentation en XML des noms propres et de leurs relations

2.3 Prolmf : Prolexbase dans le standard des ressources lexicales multingues LMF

La motivation pour concevoir un schéma XML de Prolexbase a toujours été de rendre cette ressource *interopérable* avec d'autres et exploitable par de nombreuses applications. La mise en ligne sur le site du CNRTL [20] témoigne de cet objectif. Pour autant, la version mise en ligne est dans un format XML différent de celui conçu dans le cadre de la thèse de Mickaël Tran [BTM05] car nous avons entretemps étudié comment rendre Prolexbase conforme à LMF [ISO08], un standard pour la modélisation des ressources lexicale du comité ISO/TC 37/SC 4 [Rom11]. Le travail de ce comité a débuté en 2003 et porte sur tout un ensemble de normes pour "les ressources terminologiques et autres ressources langagières". Etant donné notre objectif d'interopérabilité pour les échanges et les collaborations, il est important pour nous de formuler d'abord dans ce cadre la ressource à partager.

La figure 2.5 résume le contexte et nos propositions : nous avons représenté le système Prolex dans le cadre de LMF dans l'objectif d'augmenter l'utilisabilité de Prolexbase pour le plus grand nombre d'applications [BM08]. Cette représentation a été enrichie fin 2012 [MB13]. Pour présenter ces propositions, je rappelle brièvement les grandes lignes de

20. http ://www.cnrtl.fr/lexiques/prolex/

LMF [ISO08], puis de la notion de catégories de données qui l'accompagne, avant de décrire Prolmf, la modélisation en LMF de Prolexbase.

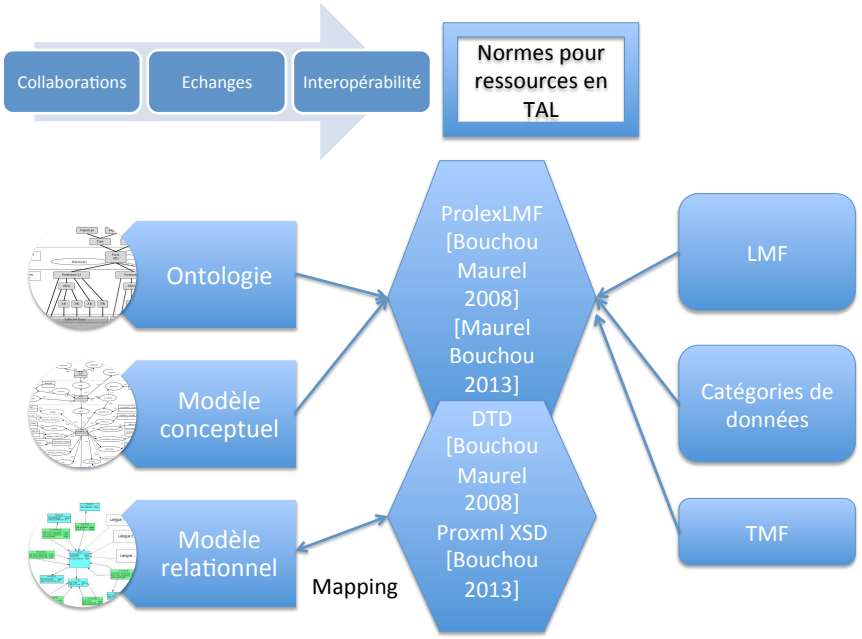


Figure 2.5 – Alignement sur les standards de représentation des ressources du TAL

2.3.1 Le modèle conceptuel LMF

LMF, norme ISO 24613, est depuis 2008 un standard pour la création et l'utilisation de ressources langagières dans un contexte monolingue et multilingue. C'est un modèle conceptuel, conçu comme un "méta modèle" destiné à être instancié en fonction des besoins [Rom10]. Il utilise la notation UML [21] pour décrire sa structure, qui comprend un noyau de composants obligatoires et un ensemble d'extensions. De plus, les termes qui qualifient les composants dans un modèle compatible LMF doivent faire partie d'un ensemble de descripteurs dont la définition relève d'une autre norme, l'ISO 12620, appelée *catégories de données* [WW12], qui fait l'objet de la section suivante.

Une ressource lexicale conforme à LMF est au moins composée d'un ou plusieurs lexiques (classe *Lexicon*), chacun contenant la description d'une langue particulière. Un lexique contient un ensemble d'entrées lexicales (classe *Lexical Entry*) qui sont composées d'un ensemble de formes (classe *Form*) et, éventuellement, d'un ensemble de sens (classe

21. 2012 edition standard : ISO/IEC 19505-1 and 19505-2 : textttthttp ://www.omg.org/spec/UML/2.4.1

Sense). Les formes peuvent être raffinées en différentes représentations (classe *Form Representation*). Quant aux sens, ils peuvent être multiples, pour représenter la polysémie, pour autant que les informations attachées aux formes soient identiques pour tous les sens.

Prenons l'exemple du mot *tour*, si nous souhaitons seulement une liste de mots nous pouvons définir une entrée lexicale contenant simplement cette forme, éventuellement avec un attribut pour sa catégorie grammaticale. Si notre lexique contient une représentation sémantique, nous pouvons ajouter à cette entrée trois sens, *construction nettement plus haute que large*, *machine-outil servant à façonner des pièces cylindriques tournant sur leur axe* et *boîte, armoire cylindrique tournant sur un pivot* (d'après le TLFI [22]). Si nous complétons encore ce mot par son genre, ce n'est plus une, mais deux entrées lexicales qu'il faudra créer, une féminine associée au premier sens et l'autre masculine, associée aux deux autres sens.

Les lexiques contenant les formes et les sens constituent le cœur de LMF, autrement dit le noyau de description obligatoire. Toutes les autres informations possibles sont réparties dans huit extensions [23]. Ainsi, pour ajouter par exemple à notre cas d'étude les deux formes fléchies *tour* et *tours*, il faut utiliser l'extension morphologique qui permet une description morphologique avec la notion de lemme (classe *Lemma*) et de formes fléchies (classe *Word Form*) [24]. Remarquons que cette extension est indépendante de la partie sémantique de l'entrée lexicale. Cette dernière aussi peut être complétée, via l'extension sémantique, avec la possibilité de définir des relations entre sens (classe *Sense Relation*) [25].

Au delà des descriptions monolingues, LMF comprend l'extension multilingue, qui permet de relier un sens propre à une langue avec un sens qui peut être partagé par plusieurs langues. On retrouve dans cette dernière extension aussi bien l'idée du pivot de traduction (classe *Sense Axis*) que celle du transfert (classes *Transfert Axis* et *Example Axis*), notions issues des principaux projets de ressources multilingues pour le traitement automatique des langues [Rom10]. Ces trois classes se rattachent directement à la ressource lexicale, comme les lexiques. Des relations entre ces pivots peuvent être définies *via* la classe *Sense Axis Relation*. De plus, la classe *Interlingual External Ref* permet de lier un pivot à des descriptions externes et la classe *Monolingual External Ref* permet également de lier un sens dans une langue à des descriptions externes (comme des *synsets* de Wordnet par exemple).

2.3.2 Les catégories de données

Les classes du modèle conceptuel LMF couvrent l'ensemble des éléments d'une ressource lexicale mais leur description passe par les catégories de données, un mécanisme qui assure la cohérence sémantique entre les diverses ressources lexicales conformes à LMF. C'est l'objet de la norme ISO 12620 [ISO09, WW12]. Les catégories de données sont elles aussi

22. textttatilf.atilf.fr/

23. *Morphology extension*, *Machine Readable Dictionary extension*, *NLP syntax extension*, *NLP semantics extension*, *NLP multilingual notations extension*, *NLP paradigm pattern extension*, *NLP multiword expression patterns extension* et *Constraint expression extension*.

24. Ainsi que les racines (classe *Stem Or Root*) ou les unités des mots polylexicaux (classe *List Of Components*).

25. Egalement des ensembles de synonymes (classe *Synset*) et des structures prédicativess (classe *Predicative Representation*), éventuellement conjointement à l'extension syntaxique.

conceptuelles, dans le sens où chacune est un concept, avec un label, un identifiant et une définition clairement documentée, ce concept étant destiné à être instancié sous la forme d'un champ de base données, d'un attribut ou d'un élément XML, etc.

Il y a 2 sortes de catégories de données, les complexes qui sont des noms de caractéristiques (/*part of speech*/, /*author*/) et les simples qui représentent des valeurs de caractéristiques (/*singular*/). Les complexes peuvent être contraintes à n'avoir de valeur que d'un certain type (numérique, chaîne de caractères) ou dans un ensemble fini de valeurs possibles. Notons que les simples restent des objets abstraits : par exemple /*singular*/ peut s'instancier en *sing* ou *s* ou *singular*...

Dans un premier abord, la norme ISO 12620 est donc un inventaire de descripteurs et de leurs valeurs, ou encore un vocabulaire normalisé pour décrire des ressources linguistiques. Elle a pour origine de grands projets sur les bases de données terminologiques (Geneter, MARTIF, etc.). Mais la description des ressources langagières est une matière vivante et si la stabilité apportée par la norme favorise les échanges, les mises en commun des données et facilite le développement d'outils pour leur étude, elle doit aussi garantir son utilisabilité à long terme en offrant des garanties d'évolutivité. C'est l'objet de la notion de registre de catégories de données (*DCR*), qui permet d'enrichir itérativement cet inventaire au fil des expériences d'utilisations. Il s'agit d'un environnement pour l'enregistrement, la documentation et la standardisation des catégories de données [BDH+08], implémenté dans la plateforme ISOcat [26].

On trouve dans cette plateforme le label et la définition de chaque catégorie de données, les valeurs qu'elle peut prendre (si elle est complexe), éventuellement les classes des modèles auxquels elle se rattache. La dimension multilingue permet de connaître les variations de sens d'une catégorie de données d'un langage à l'autre, ainsi que les variations de termes utilisés pour celle-ci. Elle repose sur une approche par pivot : le concept que représente la catégorie de donnée a un identifiant unique (une URI de la forme `http://www.isocat.org/datcat/DC-207`) par lequel on obtient la description globale et, dans un deuxième niveau, des descriptions adaptées pour divers langages.

Etant donné le parti pris collaboratif de l'environnement ISOcat, rassemblant des contributeurs qui relèvent de différents grands projets, il se trouve que le nom d'une catégorie de donnée ne correspond souvent pas à un seul pivot, c'est-à-dire à une seule URI. Par exemple, /*partOfSpeech*/ correspond aux DC-1345, DC-396, DC-3747 et DC-3748. Pour cette raison, il est recommandé d'associer explicitement le numéro de la catégorie de donnée qui correspond le mieux à l'usage que l'on souhaite au nom utilisé pour cette catégorie de donnée dans notre ressource.

Notons que pour plus d'interopérabilité un autre outil est en cours de conception pour relier des définitions de l'ISOcat (par exemple par une relation de type *sameAs*, mais pas seulement) et également de relier des définitions de l'ISOcat avec celles d'autres ressources comparables, réalisées dans d'autres cadres de description : cet outil qui sera bientôt disponible en accès public s'appelle RELcat [Win12].

La définition d'un modèle pour les noms propres dans le cadre de LMF revient donc (1) à utiliser le noyau obligatoire, (2) à choisir les extensions nécessaires et (3) à déterminer l'ensemble des catégories de données (avec leur URI précise) qui permettent de décrire les

26. http ://www.isocat.org

prolexèmes et leurs relations.

2.3.3 Le modèle conceptuel Prolmf

Prolmf est lui-même un modèle conceptuel, dédié à la description des noms propres et de leurs relations, qui a pour "instances", ou correspondants au niveau logique, Prolexbase (base de données relationnelle) et son exportation en XML. Nous l'avons décrit dans [BM08] et [MB13] à travers de nombreux exemples. La figure 2.6 schématise ses classes, et la table 2.2 met en évidence les liens entre les principes ontologiques de Prolexbase, illustrés en figure 2.1, et les classes de LMF utilisées pour les exprimer.

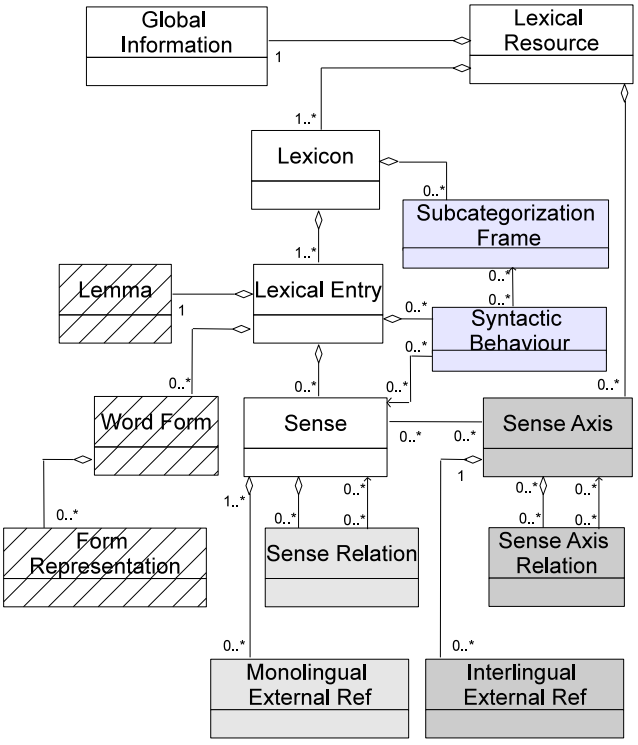


FIGURE 2.6 – Les classes de Prolmf

On y voit d'abord le noyau obligatoire de LMF : ce sont les classes sans signe distinctif dans la figure 2.6, à savoir *Lexical Resource* qui represente la ressource entière, *Global Information* qui est un en-tête contenant des méta informations sur la ressource, *Lexicon* qui représente un ensemble d'entrées lexicales (classe *Lexical Entry*) dans une langue, chacune pouvant être décrite par un ou plusieurs sens (classe *Sense*).

Pour les formes, Prolexbase est destiné à contenir à la fois des règles de flexion (mono et polylexicale) et l'ensemble des instances. Pour les lemmes et les formes fléchies nous

utilisons donc aussi les classes *Lemma*, *Word Form* et *Form Representation* de l'extension morphologique (hâchurées dans la figure 2.6) [27].

Niveaux Ontologie	**Prolexbase**	**LMF**
Indépendant des langues Méta-conceptuel	Types (taxonomie des types) Existence	Interlingual External Ref (ressources externes)
Indépendant des langues Conceptuel	Pivots	Sense Axis
	Relations	Sense Axis Relation
Dépendant des langues Linguistique	Langue	Lexicon
	Prolexème	Sense
	Alias, dérivés	Sense Relation
	Précisions des contextes syntaxiques	Syntactic Behaviour Subcategorization Frame
Dépendant des langues Instances	Formes fléchies Descriptions morphologiques	Lexical Entry, Word Form Form Representation

Table 2.2 – Résumé des correspondances Prolexbase → LMF

Pour les sens, Prolexbase distingue les relations qui ne dépendent pas de la langue (synonymie, méronymie et accessibilité) et les relations qui en dépendent (aliasisation, dérivation morphosémantique, expansion classifiante, éponymie). Il s'agit de ne pas dupliquer dans chaque langue une même information. C'est pourquoi nous utilisons les classes de l'extension multilingue (*Sense Axis*, *Sense Axis relation* et *Interlingual External ref* en gris dans la figure 2.6) pour les premières et deux classes de l'extension sémantique (*Sense Relation* et *Monolingual External Ref* en gris pâle dans la figure 2.6), plus deux classes de l'extension syntaxique (*Syntactic Behaviour* et *Subcategorization Frame* en bleu pâle dans la figure 2.6) pour les secondes.

Les tables 2.3 et 2.4 précisent les catégories de données de Prolmf, pour les classes de la figure 2.6 : celles notées entre slashs sont des entrées du registre ISOcat. La troisième colonne précise les valeurs, entre parenthèses on indique si les valeurs sont fixées par un standard ou forment un ensemble fini déterminé (fermé), quand on ne peut les énumérer toutes [28].

Force est de constater que déterminer les catégories de données pour Prolmf (trouver celles qui existent déjà, proposer celles qui devraient exister) a été et reste une question délicate étant donné l'état de l'ISOcat, mais d'une part celui-ci évolue régulièrement et d'autre part le partage du vocabulaire employé pour les descriptions est absolument indispensable pour l'interopérabilité recherchée. Le jeu en vaut donc la chandelle. Un autre point de discussions récurrentes est la sérialisation en XML du modèle, ici de Prolmf mais la question se pose pour tout modèle respectant LMF. C'est l'objet de la section suivante.

27. Les règles de flexion ne sont pas intégrées dans Prolmf.
28. Certaines valeurs entre guillemets ont été proposées comme catégories de données mais ne figurent pas encore dans le DCR.

Classes	**Catégories de données**	**Valeurs**
Lexical Resource		
Global Information	/*entrySource*/ DC-207	"Prolexbase"
	/*languageCoding*/ DC-2008	"utf-8"
	/*resourceName*/ DC-2544	"Prolmf"
	/*version*/ DC-2547	"1.2"
Lexicon	/*languageIdentifier*/ DC-279	"fra" (ISO639-3)
	/*script*/ DC-1855	"Latin-1" (Unicode)
Lexical Entry	/*partOfSpeech*/ DC-1345	/*noun*/ DC-1333
Lemma	/*writtenForm*/ DC-1836	chaîne de car.
	/*script*/ DC-1855	"Cyrillic" (Unicode)
	/*orthographyName*/ DC-2176	"withDots"
Word Form	/*writtenForm*/ DC-1836	cf. Lemma
	/*script*/ DC-1855	cf. Lemma
	/*orthographyName*/ DC-2176	cf. Lemma
	/*grammaticalNumber*/ DC-1298	/*singular*/ DC-1387 (DCR)
	/*grammaticalGender*/ DC-1297	/*feminine*/ DC-1880 (DCR)
	/*grammaticalCase*/ idem DC-1840	(DCR)
	/*grammaticalTense*/ DC-1286	(DCR)
	/*grammaticalMood*/ idem DC-366	(DCR)
	/*grammaticalPerson*/ idem DC-1328	(DCR)
Form Representation	/*writtenForm*/DC-1836	cf. Lemma
	/*script*/ DC-1855	cf. Lemma
	/*orthographyName*/ DC-2176	cf. Lemma

TABLE 2.3 – Catégories de données : ressource, lexique, formes des entrées lexicales

2.4 Schéma XML de Prolmf

J'ai abordé la conception XML dans le cadre suivant : (i) les utilisateurs sont des experts du traitement automatique des langues naturelles qui ont à constituer des ressources de descriptions des phénomènes langagiers ; (ii) ils ont développé de longue date des modèles pour représenter les informations de ce domaine ; (iii) ils ont développé de nombreux outils pour traiter ces modèles ; (iv) ils ont, pour certains, l'ambition de travailler les uns avec les autres sur les données des uns et des autres ; et (v) ils ont adopté le format XML. Aussi je présente d'abord l'existant en termes de formats XML pour des données conformes à LMF, puis je justifie nos choix pour le schéma XML de Prolmf. Je donne finalement les grandes lignes du cadre actuel pour développer un schéma XML pour des données conformes à LMF, avant la conclusion générale du chapitre.

Classes	Catégories de données	Valeurs
Sense	/*termProvenance*/ DC-509	/*fullForm*/ DC-321
	/*frequency*/ DC-1965	/*commonlyUsed*/ DC-1984 (DCR)
Sense Relation	/*label*/ DC-1857	"derivative", "alias"
Monolingual External Ref	/*externalSystem*/ DC-1974	"wikipedia"
	/*externalReference*/ DC-1975	"http ://..."
Syntactic Behaviour		
Subcategorization Frame	/*introducer*/ DC-2245	"determiner" "locativePreposition" "classifyingContext" "accessibilityContext"
	/*grammaticalGender*/	cf. DC-1297
	/*grammaticalNumber*/	cf. DC-1298
	/*restrictionRank*/ idem DC-1862	2 : partie du lemme concernée
Sense Axis		
Sense Axis Relation	/*label*/ DC-1857	"partitiveRelation" "quasiSynonymie" "associativeRelation"
	/*subjectField*/ DC-489	"capital" (fermé)
	/*usageNote*/ DC-526	"diaphasic" "diachronic" "diastratic"
Interlingual External Ref	/*externalSystem*/ DC-1974	"typology", "existence"
	/*externalReference*/ DC-1975	"city" (fermé)

Table 2.4 – Descriptions des sens dans Prolmf, au niveau d'une langue et indépendamment des langues

2.4.1 Des schémas XML proposés pour LMF

2.4.1.1 La DTD qui accompagne LMF

Les principes de LMF indiquent qu'un modèle exprimé dans ses termes peut être instancié en XML en utilisant le schéma que l'on veut pourvu qu'il soit "isomorphe" au modèle. Une proposition de DTD est contenue dans la norme elle-même : elle est assez largement adoptée dans les projets qui utilisent LMF (cf. [Fra13]), alors même qu'elle n'y est présentée qu'à titre informatif. Nous rappelons d'abord les particularités de ce schéma.

Les classes y sont représentées par des éléments XML et les relation d'héritage et d'aggrégation (vues comme des compositions) sont représentés par l'inclusion des éléments XML dans l'élément représentant la classe mère ou la classe contenante. C'est une démarche de passage d'un modèle UML (ou plus généralement un modèle conceptuel) à un schéma XML parmi d'autres, comme nous l'avons montré en section 2.1, mais les choix pour chaque association ne sont pas argumentés et il apparaît qu'il pourrait en être autrement sans trahir

le modèle LMF. Ainsi pour Prolexbase nous représentons différemment l'association entre Sense et Sense Axis, par exemple (ceci est discuté lors de la présentation de notre schéma).

L'ensemble des caractéristiques des classes LMF est instancié par le biais d'un élément générique *feat* muni de 2 attributs *att* et *val*, le premier recevant pour valeur le nom de la caractéristique (par exemple *part of speech*) et le deuxième sa valeur (par exemple *verb*), comme illustré ci-après. Etant donné que LMF représente le cadre dans lequel les projets peuvent représenter leur ressource, il ne peut en être autrement *au niveau de LMF*, le vocabulaire effectif dépendant du projet.

```
<!ELEMENT feat EMPTY>
<!--att=constant to be taken from the DataCategoryRegistry-->
<!--val=free string or constant to be taken from the DCR-->
<!ATTLIST feat
              att       CDATA  #REQUIRED
              val       CDATA  #REQUIRED>
```

Du point de vue des recommandations en conception XML, il aurait été plus correct de ne mettre que le nom *att* en attribut et la partie *val* en contenu de l'élément [29] car (i) le nom de la propriété est une information atomique sur le contenu de l'élément, alors que (ii) la valeur peut nécessiter plusieurs descriptions supplémentaires, comme par exemple son unité de mesure.

Du point de vue de la modélisation linguistique, les descriptions par structures de traits (feature structures) sont très couramment utilisées, le principe étant de décrire un objet via un regroupement de caractéristiques. Cette description est *par essence partielle* (et peut donc être complétée à tout moment). Ce type de description fait l'objet de l'ISO24610-1 :2006, qui fixe un format XML pour l'échange de telles descriptions entre différentes applications. Cette norme recommande plutôt le format qu'utilise l'exemple suivant :

```
<fs>
   <f name="orth"> <string>love</string> </f>
   <f name="pos"> <symbol value="noun"/> </f>
<fs>
```

Ce cadre est donc grosso-modo repris par la DTD qui accompagne LMF : des éléments *feat* peuvent être ajoutés partout. Pour autant, cela n'oblige pas une instance XML d'une ressource conforme à LMF à suivre ce choix de représentation XML, du moment que la contrainte de prendre les noms (et autant que possible les valeurs) dans le répertoire des catégories de données ISOcat soit bien respectée. En effet, pour un projet précis il est possible de fixer un ensemble de noms d'attributs (ou éléments) XML pour caractériser les contenus de la ressource. Du point de vue XML cela a plusieurs avantages, d'abord celui de la lisibilité, également celui de pouvoir s'appuyer sur les outils standards XML de vérification automatique de contraintes de structure, d'occurrence et de type des valeurs. Si besoin pour des échanges, il est toujours possible de se ramener au format <feat att= val=> en appliquant une transformation XSLT.

Un autre point concernant la DTD fournie avec LMF et inhérent au fait de présenter un cadre le moins contraignant possible, destiné à être instancié différemment selon les

29. Comme il est fait dans le schéma abstrait GMT accompagnant TMF, fourni comme format canonique d'échange entre des langages conformes à TMF.

projets, est que presque rien n'est obligatoire dans le schéma. Pour les mêmes raisons que pour l'élément *feat* il est raisonnable de ne pas s'en tenir à cela pour un projet particulier, car préciser le cadre permet de garantir l'intégrité des informations contenues, via des vérifications automatiques standards, sans avoir à re-développer de façon ad hoc des programmes pour réaliser de telles vérifications.

Rappelons encore que les relations entre les classes LMF ont été retranscrites en imbriquant les éléments et en utilisant le mécanisme des ID/IDREFS, ceci en choisissant un sens qui ne peut pas être celui qui convient le mieux dans tous les cas comme nous l'avons déjà pointé. Or l'intégrité référentielle ainsi recherchée est mal réalisée par les ID/IDREF(S) des DTDs [30] : à partir du moment où on a besoin de cette fonctionnalité il vaudrait mieux utiliser les langages XML Schema ou RELAX-NG avec Schematron.

Dans le même ordre d'idée, la DTD fournie avec LMF est bien découpée par modules du modèle mais cette modularité est toute relative puisque c'est une DTD. Pour suivre les recommandations de modularité classiques en conception XML les langages XML Schema ou RELAX-NG, qui permettent d'utiliser le principe des espaces de noms, sont également plus indiqués. Or la modularité est la base des possibilités de redéfinition, d'extension et de composition, qui garantissent l'évolutivité nécessaire des modèles. Par exemple pour Prolexbase, il reste des aspects qu'il a été prévu de représenter mais qui ne le sont pas encore, comme par exemple les règles de flexions et les noms composés, sans compter que l'ajout de nouvelles langues peut toujours nécessiter des descriptions qui n'auront pas été définies pour les langues déjà présentes.

En résumé, la DTD annexée à LMF ne pouvait être qu'un cadre et devrait toujours n'être considérée que comme tel : un cadre à instancier avec précision dans chaque projet se conformant à LMF et pour lequel il faut un schéma. Cependant, en tant que cadre générique elle est limitée car trop figée, peu modulaire, peu extensible, etc. Alors qu'il existe un cadre de définition de schémas XML particulièrement efficace par ailleurs, destiné au domaine général du traitement des documents textuels : c'est la TEI (Text Encoding Initiative) [31]. Alternativement, LMF commence à être adopté pour des projets de taille conséquente, en termes de quantité de données mais surtout en termes de quantité de sources distinctes à faire collaborer, et cela a suscité récemment de nouvelles propositions de schémas pouvant également jouer ce rôle de cadre générique à partir duquel produire un schéma concret.

2.4.1.2 Autres options : TEI, schémas RELISH LMF et UBY-LMF

La TEI est un cadre de développement de schémas XML pour des ressources langagières. Ce cadre permet de réutiliser les définitions de centaines d'éléments et d'attributs, mais aussi un format appelé ODD (One document does it all) et un outil pour générer une DTD, un schéma en XML Schema ou un schéma en RelaxNG, à partir d'un fichier contenant un ODD précis : Roma [32]. La TEI contient un chapitre pour l'annotation des entités nommées rencontrées dans des textes, évidemment très restreint par rapport au contenu de Prolexbase.

30. Comme souligné dans le chapitre 1.
31. `http://www.tei-c.org/release/doc/tei-p5-doc/`
32. `http://tei.oucs.ox.ac.uk/Roma/`

Dans [Rom10, Rom11, RW12, HFR12], il est montré qu'en utilisant les éléments structurants <entry>, <form>, <gramGrp> et <sense> du chapitre *Dictionnaires* de la TEI, il est possible de représenter directement les classes Lexical Entry, Form, Form Representation et Sense de LMF. Ainsi sont couvertes les descriptions morphologiques et grammaticales d'un lexique de formes fléchies et des précisions syntaxiques peuvent également être décrites [HFR12]. Il est montré de plus que la description des sens dans un tel lexique, avec les particularités qui leur sont liées (définitions <def>, exemples, notes d'usage <usg>, etc.) est également réalisable et le potentiel de l'élément récursif <cit> de la TEI est mis en évidence, en particulier pour des dictionnaires bilingues : un sens contenant un <cit> introduit l'entrée correspondante dans l'autre langue [RW12].

Pour autant il n'est pas donné de piste pour représenter l'ensemble des classes de tous les packages LMF, en particulier les extensions syntaxique, sémantique et multilingue qui servent à définir Prolmf. Ainsi, comment représenter les Sense Axis et leurs relations, fondamentaux pour Prolmf ? Dans la forme bilingue proposée dans [RW12], la traduction d'une langue à l'autre est représentée par un élément <cit> et les éléments des deux langues coexistent dans un même "lexique", comme c'est le cas également dans Global Atlas [FMCP13]. Dans ce dernier il s'agit de modélisation "à la Wordnet", avec le synset qui se charge de regrouper les sens. En attachant un "sens" différent à chaque occurrence d'une entrée lexicale dans une langue différente, le lien est fait par le synset. Que l'élément <cit> puisse jouer un rôle similaire n'est pas immédiat. Pour Prolmf, il faudrait vraisemblablement utiliser des éléments du chapitre *Liens, Segmentation et Alignement* de la TEI pour réaliser les Sense Axis et plus généralement le niveau multilingue, mais il n'y a rien de directement utilisable, comparable à ce qui existe pour le noyau de LMF dans le chapitre *Dictionnaires*.

La TEI est une réalisation phare du domaine du traitement *des documents*. Certains de ses concepteurs sont de ce courant "document" qui, en collaboration avec le courant "bases de données", a donné naissance à la définition de XML. Il est notoire que les points de vue "document" et "bases de données" amènent à des représentations XML différentes. Au niveau des standards, les langages de définition de schémas RELAX-NG et XML Schema et plus encore, les langages d'interrogation et de transformation XSLT et XQuery reflètent les différences entre ces deux points de vue, complémentaires. Les ressources lexicales décrites par LMF représentent pour la plupart des ensembles d'informations, de descriptions : ce sont des *bases de données* lexicales. C'est peut-être là que réside la difficulté qu'il y a à aller plus loin que la représentation d'un lexique de formes fléchies à l'aide de la TEI, de façon simple et intuitive. Un article tout récent [BMM12] sur une expérience de l'ICLTT [33] d'une part pointe le manque de références en la matière et d'autre part décrit comment leurs dictionnaires sont désormais décrits dans le cadre de la TEI. Ils expliquent qu'ils ont également étudié le potentiel de LMF et qu'ils restent attentifs à ses développements mais qu'ils l'ont considéré encore trop peu validé dans de grands projets. Les choix qu'ils exposent peuvent néanmoins inspirer de nouvelles pistes pour bâtir un cadre TEI complet pour une sérialisation des modèles LMF en XML.

Deux projets d'envergure s'appuyant sur LMF, tout à fait récents, viennent combler le manque énoncé dans le dernier article cité (ICLTT) : RELISH et UBY. Pour l'un comme

33. Institute for Corpus Linguistics and Text Technology of the Austrian Academy of Sciences

pour l'autre un schéma XML plutôt générique a été conçu et validé (ou est en cours de validation) sur plusieurs ressources. Le projet RELISH [ADDWN12, WPN+13] a pour objectif de rendre interopérables des ressources lexicales développées pour des langues naturelles pour lesquelles il y a peu de ressources, parfois en voie de disparition. Ce projet adresse en particulier les divergences entre des approches utilisant LMF et ISOcat d'une part et d'autre part d'autres approches utilisant un format XML appelé LIFT [Hos06], développé au sein du projet LEGO de l'université Eastern Michigan, et l'ontologie de descripteurs linguistiques GOLD [FL03]. Pour permettre des échanges entre les ressources développées dans ces deux cadres différents, un schéma XML en RELAX-NG a été conçu aussi proche que possible de la DTD en annexe de LMF, mais qui pallie à quelques unes de ses limites. Il étend également cette DTD pour y intégrer (en options) des descriptions existant dans LIFT mais absentes de LMF. Ce schéma est appelé RELISH LMF et est disponible en ligne [34]. Sa publication en ligne est accompagnée de précisions sur les choix effectués en matière de conception XML pour respecter au mieux le cadre LMF. Il n'a été publié qu'en 2012, pour l'instant il n'a pas encore été testé dans le cadre d'autres projets que RELISH. C'est le cas également pour UBY-LMF, introduit ci-après.

Le projet UBY [GEKH+12] a pour objectif d'unifier concrètement la représentation de nombreuse ressources lexicales conçues avec des points de vue différents. Pour cela a été conçu UBY-LMF [EKGH+12], un modèle LMF pour des ressources lexico-sémantiques multilingues et hétérogènes, à grande échelle. UBY-LMF permet de standardiser dans ces ressources les informations lexicales à un niveau fin, en utilisant un grand nombre de catégories de données de l'ISOCat. Ce modèle a été validé par la conversion de neuf ressources en deux langues (Anglais et Allemand) vers le format XML correspondant, lequel est actuellement une DTD librement téléchargeable [35]. Les neufs ressources en question sont d'un volume conséquent : il s'agit de WordNet, ainsi que des versions anglaises de Wiktionary, Wikipedia, OmegaWiki, FrameNet et VerbNet et pour l'Allemand, Wikipedia, Wiktionary, et GermaNet [EKGH+12]. Ces versions sont désormais interopérables et peuvent être interrogées via une API Java publique commune [36]. UBY-LMF couvre un large éventail de types d'information et inclut également des liens entre les différentes ressources au niveau du sens des mots. Le modèle et le format XML ont été conçus dans l'objectif d'accueillir de nouvelles ressources et de nouvelles langues. Cet aspect en fait de facto un cadre générique.

Notre développement du schéma XML de Prolmf a démarré en 2007, alors que la seule référence était la DTD annexée à la norme : comme les deux projets précédents (RELISH et UBY), nous sommes partis de cette DTD et nous l'avons adaptée et précisée pour les besoins de notre ressource, en veillant également à prendre en compte les démarches de conception classiques adaptées à XML, présentées au début de ce chapitre. Toutefois nous nous sommes concentrés uniquement sur Prolmf aussi notre proposition ne saurait être considérée comme un cadre générique, mais bien comme une instance concrète de schéma XML conforme à LMF.

34. `http://tla.mpi.nl/relish/lmf/`
35. `http://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/uby-lmf/`
36. `http://code.google.com/p/uby/`

2.4.2 Le schéma XML de Prolmf

Le schéma XML actuel de Prolmf est décrit dans un XML Schema Document (XSD), fourni sur le site du CNRTL (voir note de bas de page numéro 20). Notons qu'il peut être complété par l'ensemble des déclarations d'associations entre les noms d'attributs (et certaines de leurs valeurs) et les URI précises des catégories de données auxquelles ils correspondent. Ce schéma XML a été conçu à la fois par "translation" depuis Prolmf et en tenant compte de considérations sur les dépendances de données, avec une dimension supplémentaire fondamentale : fonder systématiquement les choix de représentation sur les besoins des utilisateurs (contributeurs et exploitants de la ressource). Il est à noter que ce schéma contient des contraintes précises, qui représentent des réalités de notre cadre concret (description multilingue des noms propres et leurs relations), formulées déclarativement pour être vérifiées automatiquement par les outils standard d'exploitation des documents XML.

Comme dans les schémas évoqués dans les sous-sections précédentes, les classes Prolmf y sont représentées par des éléments XML portant leurs noms. Nos choix de représentation des associations entre les classes de Prolmf sont par contre propres à notre cadre. La plupart des cas d'agrégation du diagramme de classes de Prolmf (figure 2.6) sont traités en fait comme des compositions : on imbrique l'élément représentant la classe agrégée dans l'élément représentant la classe agrégeante. C'est le cas par exemple des associations (1) entre Lexical Resource et Global Information (2) entre Lexical Resource et Lexicon et (3) entre Lexical Resource et Sense Axis, qui aboutissent à la définition du type Lexical-Resource_type associé à l'élément (global) LexicalResource reproduit ci-dessous. Un tel élément contient obligatoirement un élément GlobalInformation et un seul, puis un certain nombre d'éléments Lexicon, puis un certain nombre d'éléments Sense Axis.

```
<complexType name="LexicalRessource_type">
    <sequence>
        <element ref="pxb:GlobalInformation"/>
        <element ref="pxb:Lexicon" maxOccurs="unbounded"/>
        <element ref="pxb:SenseAxis" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

Le cas des associations entre Sense et Sense Relation est matière à plus de réflexion : les sens des entrées lexicales qui correspondent au même prolexème référencent le même Sense Axis, c'est ainsi qu'ils sont reliés (cf. table 2.2). Ainsi, le sens *capitale de la France* de l'entrée lexicale *Paris* et un sens de l'entrée lexicale *Parisien* font référence au même Sense Axis. De plus, comme le montre la figure 2.1, les éléments d'un même prolexème sont liés par une relation d'alias ou de dérivé. C'est un lien hiérarchique (cf. figure 2.1) : la forme canonique complète du nom propre (par exemple *Organisation des Nations Unies*) peut avoir des alias et des dérivés, les alias peuvent avoir des dérivées (*onusien* pour l'alias *ONU*) et (en serbe par exemple) les dérivés peuvent également avoir des dérivés. La classe Sense Relation de Prolmf représente ce lien.

Dans le schéma XML, cette classe peut devenir un élément SenseRelation imbriqué dans l'élément Sens, qui doit pouvoir référencer un autre sens. Pour cela on ajoute un identifiant (attribut *id*) à l'élément Sense. Cet élément SenseRelation précise le sens en indiquant le

fait d'être un alias ou un dérivé (attribut *label*), de quoi (attribut *refSense*) et de quel type (attribut *termProvenance*). Il n'y a qu'un seul sous-élément SenseRelation par Sense car il représente une arborescence (un seul parent dans la hiérarchie de dérivation).

Plusieurs remarques peuvent être faites :

- Du fait qu'il n'y a qu'une occurrence de ce groupe d'information par élément Sense et que ce groupe d'information est un ensemble d'attributs, il est également possible de *placer ces attributs directement dans l'élément Sense.*
- L'attribut *label* est implicitement contenu dans l'attribut *termProvenance* : si la valeur de *termProvenance* est "abbreviation" c'est un alias, si c'est "RelationalName" c'est un dérivé, etc. Cet attribut *label* peut être optionnel.

C'est pourquoi dans l'extrait du schéma ci-après, on représente la classe SenseRelation par les attributs *refSense*, *label* (optionnel) et *termProvenance* placés dans l'élément Sense. On ajoute également à l'élément Sense un attribut *id*.

```
<complexType name="Sense_type">
    <sequence>
        <element ref="pxb:SyntacticBehaviour" minOccurs="0"
            maxOccurs="unbounded"/>
        <element ref="pxb:MonolingualExternalRef" minOccurs="0"
            maxOccurs="unbounded"/>
    </sequence>
    <attribute name="refSenseAxis" type="integer" use="required"/>
    <attribute name="id" type="integer" use="required"/>
    <attribute name="refSense" type="integer" use="required"/>
    <attribute name="label" use="optional">
        <simpleType>
            <restriction base="string">
                <enumeration value="alias"/>
                <enumeration value="derivative"/>
            </restriction>
        </simpleType>
     </attribute>
     <attribute name="termProvenance" use="required">
            <simpleType>
                <restriction base="string">
                    <enumeration value="fullForm"/>
                    ...
                </restriction>
            </simpleType>
     </attribute>
     <attribute name="frequency" use="required">
            <simpleType>
                <restriction base="string">
                    <enumeration value="infrequentlyUsed"/>
                    <enumeration value="rarelyUsed"/>
                    <enumeration value="commonlyUsed"/>
                </restriction>
            </simpleType>
     </attribute>
</complexType>
```

Le cas des associations entre Lexical Entry, Syntactic Behaviour et Subcategorization Frame est également intéressant : le rôle de la classe Syntactic Behaviour de Prolmf est de représenter un ensemble de précisions syntaxiques concernant une entrée lexicale d'un prolexème. Par exemple on peut préciser qu'en français on dit *la* France (déterminant) et *en* France (préposition locative). Cet ensemble de descriptions se rapporte à un sens précis (on aura des valeurs différentes s'il s'agit du paquebot France). Les descriptions elles-mêmes sont réalisées par des instances de la classe Subcategorization Frame. Il peut arriver que deux sens aient la même description [37], dans ce cas pour chacun des deux sens une instance de Syntactic Behaviour référencera la même description.

Étant donnée cette analyse, dans le schéma XML la classe Syntactic Behaviour est représentée par un élément SyntacticBehaviour imbriqué dans l'élément LexicalEntry, pas directement au premier niveau mais sous l'élément Sense. Cet élément SyntacticBehaviour contient (en attribut) une référence à une description syntaxique qui, en XML, est un élément SubcategorizationFrame imbriqué sous l'élément Lexicon. De cette manière deux sens d'une même entrée lexicale ou de deux entrées lexicales distinctes peuvent référencer la même description syntaxique. Comme expliqué dans [MB13], on ajoute également dans l'élément SyntacticBehaviour un attribut (optionnel) pour référencer une relation entre pivots (élément SenseAxisRelation) dans le cas d'un contexte d'accessibilité comme "capitale de la" dans *Paris* ***capitale de la*** *France*. L'extrait de document XML ci-dessous montre une partie des entrées lexicales pour France et Paris.

```
<LexicalEntry partOfSpeech="noun">
  <Lemma>France</Lemma>
  <WordForm grammaticalGender="feminine" grammaticalNumber="singular">
      France</WordForm>
  <Sense refSenseAxis="27" termProvenance="fullForm" frequency="
      commonlyUsed">
        <SyntacticBehaviour refSubcategorizationFrame="collocation_3"/>
        <SyntacticBehaviour refSubcategorizationFrame="collocation_7"/>
        <MonolingualExternalRef externalSystem="Wikipedia"
            externalReference="http://fr.wikipedia.org/wiki/France"/>
  </Sense>
  ...
</LexicalEntry>
<LexicalEntry partOfSpeech="noun">
  <Lemma>Paris</Lemma>
  <WordForm grammaticalGender="masculineFeminine" grammaticalNumber="
      singular">Paris</WordForm>
  <Sense refSenseAxis="38558" termProvenance="fullForm" frequency="
      commonlyUsed">
        <SyntacticBehaviour refSubcategorizationFrame="collocation_1"/>
        <SyntacticBehaviour refSubcategorizationFrame="
            classifyingContext_222"/>
        <SyntacticBehaviour refSenseAxisRelation="1"
            refSubcategorizationFrame="accessibilityContext_4"/>
        ...
  </Sense>
```

37. Par exemple *Paris* France et *Paris* Texas sont toutes deux des villes et le contexte classifiant "la ville de" sera valable pour les deux.

```
  ...
</LexicalEntry>
...
<SubcategorizationFrame id="collocation_1" introducer="determiner">zero
    </SubcategorizationFrame>
<SubcategorizationFrame id="collocation_3" introducer="determiner">la</
    SubcategorizationFrame>
<SubcategorizationFrame id="collocation_7" introducer="
    locativePreposition">en</SubcategorizationFrame>
<SubcategorizationFrame id="classifyingContext_222" introducer="
    classifyingContext" grammaticalGender="feminine" grammaticalNumber=
    "singular">la ville de </SubcategorizationFrame>
<SubcategorizationFrame id="accessibilityContext_4" introducer="
    accessibilityContext" grammaticalGender="feminine"
    grammaticalNumber="singular">la capitale de la </
    SubcategorizationFrame>
...
```

Considérons pour finir le cas de l'association entre Sense et Sense Axis : la DTD en annexe de LMF met dans l'élément Sense Axis un attribut de type liste de références à des éléments Sense, mais cela ne correspond pas à la sémantique de l'association entre Sense et Sense Axis dans Prolmf. Un Sense Axis représente un point de vue sur un référent, qui correspond dans chaque langue à un prolexème, qui consiste lui-même en un ensemble d'entrées lexicales (cf. figure 2.1). S'il fallait représenter cela dans l'élément Sense Axis une liste de références ne suffirait pas, il faudrait au moins grouper les références par langue. Par ailleurs, lister l'ensemble des éléments du prolexème au niveau du pivot, ceci pour chaque langue, introduit de la redondance d'information puisque ces éléments doivent surtout être reliés entre eux dans le lexique. De plus, il manquerait un moyen d'accéder au pivot depuis le prolexème : cet accès ne peut être réalisé qu'à partir des éléments Sense.

Pour ces raisons, dans notre schéma XML, l'association entre Sense et Sense Axis est réalisée de la façon suivante : chaque composant d'un prolexème fait référence depuis son élément Sense au Sense Axis représentant le pivot du prolexème (c'est l'attribut *refSenseAxis* qui apparaît dans l'extrait de document XML précédent). De cette manière, pour accéder à un composant du prolexème correspondant à un pivot P dans une langue donnée, il suffit de chercher dans le lexique de la langue une entrée lexicale qui a un sens dont l'attribut *refSenseAxis* a la valeur de P. Si on cherche un élément du prolexème français du pivot 27, on pourra trouver l'entrée lexicale *France*. On saura que c'est la forme canonique de ce prolexème car ce même sens a "*fullForm*" pour valeur de son attribut *termProvenance*. Pour avoir tous les composants du prolexème il suffit de sélectionner toutes les entrées lexicales qui ont un sens dont l'attribut *refSenseAxis* a la valeur 27, etc. Inversement, l'accès au pivot depuis n'importe quel composant du prolexème passe simplement par cet attribut *refSenseAxis* de l'élément Sense.

Ainsi nos choix de représentation en XML des associations entre les classes du modèle LMF ne correspondent pas tous à ceux réalisés dans la DTD annexée à LMF, ni au schéma RELISH LMF (qui reprend tous les choix faits pour la DTD à ce niveau). Ils sont pourtant justifiés dans notre contexte, pour respecter les caractéristiques de Prolexbase.

Dans LMF, selon le point de vue semasiologique, l'entrée lexicale réalise le lien entre les

composants descriptifs Form et Sense : les différents sens possibles d'une même forme sont rassemblés sous l'entrée lexicale. Inversement, selon le point de vue onomasiologique suivi pour les ressources terminologiques et formalisé dans le cadre TMF [ISO03], la description linguistique part du concept et regroupe autour de celui-ci l'ensemble des formes qui l'expriment. Prolexbase allie les deux points de vue, semasiologique et onomasiologique, puisqu'il est important de pouvoir entrer dans la ressource via le mot, mais ensuite il est particulièrement important de pouvoir naviguer dans la ressource via le sens :

- Pour les applications en recherche d'information, au sein d'une même langue : quels sont les mots liés à ce nom nom propre, soit appartenant au même prolexème, soit via les relations sémantiques indépendantes de la langue ?
- Pour les applications liées à la traduction : quel est l'équivalent dans la langue cible, s'il n'existe pas directement comment le construire via les éléments du même prolexème, etc.

Prolexbase est donc sans conteste une ressource lexicale, qui décrit les mots, leurs formes et caractéristiques, puis leur sens, mais en plus, son organisation décrite en figure 2.1 est également proche de celle d'une ressource terminologique, du fait que l'essentiel de la description sémantique est réalisé au niveau indépendant des langues. Exprimé dans le cadre LMF, Prolmf respecte l'organisation semasiologique et en offre donc les avantages : nous avons préservé cet aspect lors du passage à XML. De plus, notre réalisation en XML, en particulier de la relation entre Sense et Sense Axis, reproduit dans sa vue XML l'originalité majeure de Prolexbase, l'articulation pivot-prolexème, en permettant à la fois (1) d'accéder directement à la description sémantique depuis un constituant d'un prolexème d'une langue donnée, et (2) de retrouver efficacement les constituants du prolexème correspondant à un pivot donné, dans chaque langue représentée. Pour autant ce schéma reste transformable vers un autre schéma conforme à LMFvia des transformations XSLT comme celles décrites dans [EM13], du fait que nous avons veillé à ce que nos choix n'entraînent pas de perte d'information propre au modèle UML de LMF.

2.4.3 Pour construire un schéma XML d'une ressource conforme à LMF

Pour avoir une base de comparaison et tirer des enseignements de ce travail de conception de schéma XML, j'ai rassemblé dans le tableau 2.5 un ensemble de projets dont les données sont conformes à LMF. Les leçons qui peuvent être tirées pour la construction d'un schéma XML d'une ressource lexicale axée sur l'interopérabilité, sont les suivantes :

- La communauté scientifique internationale concernée a travaillé activement depuis une dizaine d'années à l'élaboration d'un cadre normatif pour guider l'élaboration des ressources lexicales, au sein du du comité ISO/TC 37/SC 4. Il est indispensable de concevoir le schéma XML en respectant ce cadre, précisément LMF et les catégories de données.
- Une fois la ressource clairement formalisée avec LMF, il faut en produire un schéma XML. Il y a pour cela plusieurs options :
  - Utiliser la DTD en annexe de la norme LMF. C'est l'option la plus immédiate, pourtant, du fait que cette DTD ne représente pas une ressource concrète mais un moule général, relativement peu adaptable, elle peut ne pas convaincre les utilisateurs, concepteurs, contributeurs et exploitants de la ressource. Par exemple, elle

Projets	Ressources	Schemas XML
Asian Language ressources [a]	ressources de différentes langues asiatiques, en particuliers Chinois, Japonais et Thai [TLS+13]	DTD de LMF (plus un modèle en OWL/RDF)
Dbnary [b]	réseaux lexical LMF extrait de Wiktionnaires [Sér12]	Sérialisation de LMF en RDF
Dialectes arabes	dictionnaires bilingues Irakien, Syrien et Marocain [DG12]	DTD de LMF
DILAF [c]	ressources de différentes langues d'Afrique de l'Ouest, en particuliers 5 dictionnaires bilingues [EKM+12, EM13]	DTD de LMF et autre schéma (plus adapté au travail des lexicographes)
DUELME [d]	lexiques de noms composés hollandais (environ 5000) [Odi13]	un schéma en Relax-NG
Global Atlas [e]	Ressource en LMF de noms propres extraits en continu de Wikipedia (plus d'1 million d'entrées lexicales) [FMCP13]	DTD de LMF
KYOTO [f]	9 dictionnaires sémantiques à la Wordnet dans 9 langues [PV13]	Wordnet-LMF (pas d'éléments *feat*)
LEMON [g]	modèle de lexiques liés au web sémantique (ontologies) [MSC11]	LEMON RDF
Lexicon-Grammar [h]	Lexique Grammaire (application au français) [LTC13]	DTD de LMF
LMF for Arabic [i]	lexiques syntaxiques HPSG arabes (environ 37000 entrées lexicales) [HFR12, KGHH13]	DTD de LMF ([KGHH13]) et TEI ([HFR12])
METANET4U [j]	correspondances et fusions de ressources en LMF [VPB13] (volumes de l'ordre de 100000 entrées lexicales)	DTD de LMF comme format pivot
Morphalou 2 [k]	dictionnaire de formes fléchies (environ 540000)	format antérieur à la DTD de LMF
Prolex [l]	Prolexbase (noms propres en français, anglais, polonais et serbe, environ 180000 entrées lexicales) [MB13]	Prolmf-XML
RELISH [m]	format pivot pour différentes ressources LMF [WPN+13]	RELISH-LMF
Services web [n]	Services web d'accès à des ressources LMF [HMS+13]	Wordnet-LMF
UBY [o]	UBY-LMF : 9 ressources lexico-sémantiques en Anglais et en Allemand [EKGH+12, EKGH+13]	DTD de UBY-LMF
U.S. Gov. lexical resources	Syst. de gestion de ressources lexicales et traduction [Mon13]	adaptation de la DTD de LMF
Wiktionaries	codage Wiktionnaires en LMF (toutes langues) [DLM13]	sérialisation XML de RDF (?), plus SKOS

Table 2.5 – Projets qui utilisent LMF

a. `http://www.cl.cs.titech.ac.jp/ALR/`
b. `http://dbnary.forge.imag.fr/`
c. `http://dilaf.org/`
d. `http://duelme.inl.nl`
e. `http://www.globalatlas.org`
f. `http://www.kyoto-project.org` et `http://www.kyoto-project.eu`
g. `http://www.lemon-model.net/`
h. `http://infolingu.univ-mlv.fr/english`
i. Laboratoire MIRACL `http://www.miracl.rnu.tn`
j. `http://www.meta-share.eu`
k. `http://www.cnrtl.fr/lexiques/morphalou/LMF-Morphalou.php`
l. http ://www.cnrtl.fr/lexiques/prolex/
m. `http://tla.mpi.nl/relish/`
n. `http://chiron.lang.osaka-u.ac.jp/scope`
o. `http://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/uby-lmf/`

contient des déclarations pour toutes les classes de tous les extensions (probablement aucune ressource concrète n'a besoin de tous les packages) mais en même temps elle ne permet d'exprimer aucune contrainte concrète sur les valeurs. Elle présente par ailleurs les limites évoquées sur le site où est présenté le schéma RELISH LMF [38] : elle s'appuie sur les traits et structures de traits mais sans en suivre la norme, elle est difficilement extensible, les définitions sont peu factorisables donc peu réutilisables, etc. Parmi les projets identifiés dans le tableau 2.5, 7 utilisent la DTD telle quelle.

- Utiliser le schéma RELISH LMF (en RELAX-NG), conçu pour résoudre certaines des limites de la DTD précédemment évoquée : c'est une solution "clé en main" raisonnable qui est modulaire, adaptable et extensible. En particulier, elle permet d'utiliser la partie de la TEI P5 définissant les structures de traits selon la norme ISO 24610, ce qui offre du même coup un moyen clair de faire référence aux numéros des catégories de données (URI). Le fait d'utiliser Schematron avec Relax-NG permet d'exprimer des contraintes, comme la relation d'une sous-classe avec une super-classe abstraite (via un attribut comme @lmf :type par exemple). Même si ses représentations en XML des associations du modèle LMF (qui suivent exactement celles la DTD) peuvent être remises en cause, ce qui est attendu des auteurs [39], ce schéma très récent sera peut-être plus utilisé à l'avenir. Il n'est pour l'instant pas utilisé en dehors du projet RELISH.
- Utiliser la DTD de UBY-LMF : pour une ressource de type lexico-sémantique comparable aux neuf qui ont été intégrées dans le projet c'est la meilleure solution. Ce format est tout aussi récent que le précédent et n'a pas non plus encore été mis en œuvre dans d'autres projets.
- Utiliser la TEI : comme démontré dans [RW12, Rom13] et décrit succinctement dans la section précédente, il est possible d'utiliser directement le chapitre Dictionnaires pour représenter des ressources lexicales conformes à LMF qui auraient besoin essentiellement du noyau principal plus des précisions sur les formes, les sens et les descriptions grammaticales. A notre connaissance, il n'y a pas de projet complet parmi ceux du tableau 2.5 qui utiliserait cette solution, mais des pistes sont évoquées dans [HFR12].
- Considérer toutes ces solutions pour en élaborer une qui reflète bien le point de vue des utilisateurs, qui aiment à retrouver intuitivement dans la représentation XML leur perception habituelle de leur ressource et, quand c'est le cas, l'adoptent plus aisément en acceptant les contraintes induites par l'objectif d'interopérabilité. Parmi les projets du tableau 2.5, 7 ont pris cette voie. Il est intéressant de noter que l'un d'entre eux ([EM13]) utilise plusieurs schémas, dont un schéma "pivot" qui est aisément compris par les lexicographes et un schéma "cible" qui correspond à la DTD annexée à LMF (avec des programmes XSLT pour passer de l'un à l'autre).

- Quelle que soit l'option choisie parmi les précédentes, le schéma ne restera pas gravé dans le marbre, il devra évoluer, peut-être même en changeant d'option, donc il faut le documenter. Cela est nécessaire également pour qu'il puisse être aisément mis en correspondance avec des schémas XML de ressources conformes à LMF construits en

38. `http://tla.mpi.nl/relish/lmf/`
39. cf. Unresolved issues where especially any advice/feedback is highly appreciated.

suivant une des autres options parmi celles évoquées précédemment.

2.5 Conclusion et perspectives

La conception de schéma XML est difficile dans la mesure où il s'agit d'un modèle *logique* de données et non d'un modèle de niveau conceptuel. Comme tout modèle logique il permet de décrire des contraintes de structures et d'intégrité destinées à être vérifiées automatiquement pour favoriser la qualité des données. En particulier des notions de formes normales ont été définies ([AL02, VLL04]) et les raisonnements connus dans le monde relationnel continuent à être étendus à XML ([FHL+12, VLM12]). Cependant ce niveau de description est déjà spécifique à un cadre théorique précis (le modèle relationnel, les arbres ou les graphes) et permet difficilement à une communauté non spécialisée en bases de données d'être efficace dans ses choix de représentation. C'est pourquoi naturellement la démarche suivie par le comité ISO/TC 37/SC 4 est de spécifier les modèles au niveau conceptuel.

Le passage du modèle conceptuel à un schéma XML étant beaucoup moins immédiat que le passage d'un modèle conceptuel à un schéma relationnel, des guides doivent être élaborés (et suivis) en fonction des applications. Un moyen particulièrement recommandé est en effet de s'inspirer des choix et des schémas réalisés dans des cadres proches et qui ont déjà fait leur preuve dans suffisamment d'applications. Dans notre étude de cas, la TEI pourrait jouer ce rôle de bonne référence. La figure 2.7 présente une des perspectives pour compléter notre quête d'interopérabilité, qui consiste à construire une autre sérialisation XML de Prolmf dans le cadre de la TEI. Cela pourrait améliorer l'interopérabilité des outils du projet Prolex pour l'extraction automatique des entités nommées [FM04, MFN+12], en particulier lorsqu'ils sont employés dans le cadre de campagnes d'évaluation [NAFM10] qui utilisent le chapitre de la TEI sur les noms propres, comme ESTER/ETAPE [40].

Car XML reste la bonne cible pour l'interopérabilité, la portabilité, l'extensibilité des données et de leur structure. Il est la structure de données des standards du web. Pour autant il en existe une autre, une recommandation du W3C apparue en 1999 et stabilisée en 2004, pour laquelle on constate une forte montée en puissance depuis la fin des années 2000. Il s'agit des graphes RDF [41], dont les éléments (nœuds et arcs) sont également définis par des schémas (en RDFS [42] ou en OWL [43]), mais cette fois à un niveau clairement conceptuel [GFZC12]. En effet, RDFS n'est pas pour RDF ce qu'un schéma XML est pour XML. Un schéma XML contraint la structure d'un document XML d'un point de vue syntaxique, tandis que RDFS permet de relier les éléments RDF à une *ontologie*, définie par des classes et des propriétés organisées par la relation d'héritage. La question de la sérialisation de ces graphes en XML est tranchée dans la définition-même de ce standard, garantissant par défaut les possibilités d'échange et d'interopérabilité au niveau syntaxique. Ainsi l'interopérabilité est réalisée au niveau des données par XML et au niveau des concepts par RDF et RDFS (ou OWL).

40. `http://www.afcp-parole.org/etape.html`
41. `http://www.w3.org/RDF/`
42. `http://www.w3.org/TR/rdf-schema/`
43. `http://www.w3.org/2001/sw/wiki/OWL`

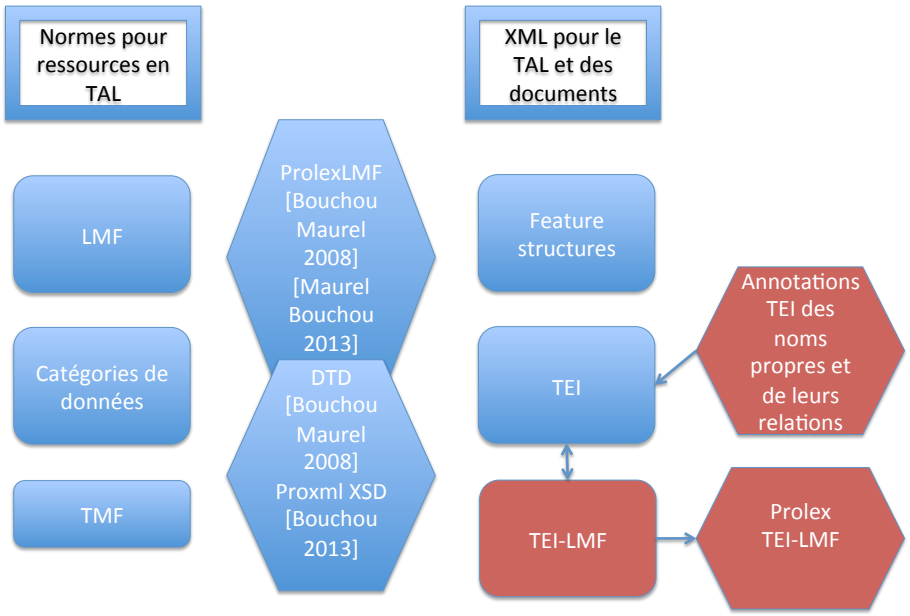


FIGURE 2.7 – Prolmf et la TEI

Gil Francopoulo rapporte dans [FMCP13] qu'au cours de la construction de la norme LMF, la question de s'appuyer sur RDF plutôt que sur XML a été débattue[44]. C'est une indication sous la forme d'une DTD qui a été finalement fournie. Cependant, on sait qu'une transcription quasi-directe des schémas UML en OWL-DL est possible (cf. travaux de Diego Calvanese et al. [BCG05, CGL+09, QACT12]). Par ailleurs, les catégories de données peuvent également être considérées dans un format sémantique, comme l'ont montré Christian Chiarcos dans [Chi10] et Menzo Windhouwer et Sue Ellen Wright dans [WW12]. Nous pourrions donc, comme l'illustre la figure 2.8, définir également une version sémantique de Prolmf. Il est intéressant de noter que les concepteurs de UBY-LMF indiquent également dans [EKGH+13] que leur système peut aussi bien être sérialisé en RDF.

Cela permettrait d'utiliser la ressource dans des applications capables de tirer partie des possibilités d'inférences offertes par ce niveau, par exemple dans un objectif d'intégration de différentes sources de données hétérogènes tant structurellement que sémantiquement, et indépendantes (voir le chapitre suivant). Plus généralement, cela permettrait d'interrelier Prolexbase aux ressources sur les entités nommées développées dans le cadre du web sémantique comme par exemple YAGO-2 [HSBW13], soit dans un réseaux pair-à-pair, soit via un annuaire centralisé dédié comme le suggère le système NERD [RTHB12].

44. Il y a une proposition de définition de LMF en OWL à l'adresse `http://www.lexicalmarkupframework.org/`.

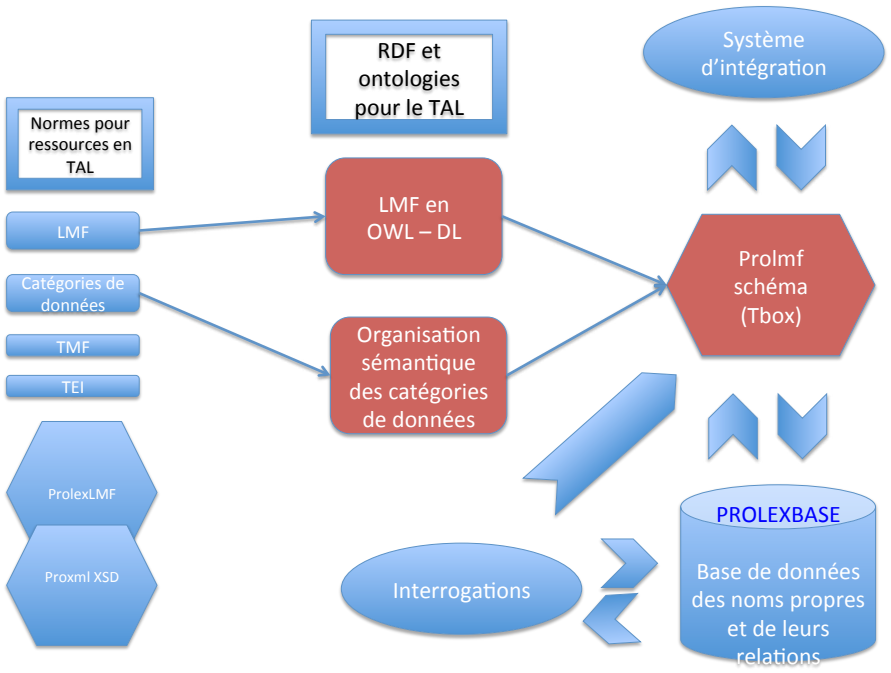


Figure 2.8 – Prolmf et le web sémantique

Plusieurs autres projets de descriptions de ressources lexicales ont déjà pris la direction du web sémantique, en s'appuyant par exemple sur des ontologies existantes pour les descriptions linguistiques comme GOLD [45] ou DOLCE [46]. Le passage à OWL/DL des ressources du projet MULTEX-East [CE11] (dont le format original est la TEI) est un autre exemple significatif, tout comme l'exploitation conjointe de ressources lexicales et d'ontologies du système LEMON [47], décrite dans [MSC11], sans compter la création du groupe de travail OWLG (Open Linguistics Working Group) de la foundation OKFN (Open Knowledge Foundation [48]), dont la vocation est décrite entre autres dans [CHN+12].

Il y a également des alternatives à la duplication des données sous prétexte de changement de format de représentation : rappelons que la ressource Prolexbase est maintenue sous la forme d'une base de données relationnelle et qu'elle est actuellement exportée dans le format ProLMF en XML. Il a récemment été proposé dans [GKK+13] un système de gestion conjointe de documents XML et de connaissances codées en RDF à propos des contenus des documents, intégrant la possibilité d'interroger conjointement ces contenus et

45. General Ontology of Linguistic Descriptions [FL03, GLS+05] : http ://linguistics-ontology.org/
46. Descriptive Ontology for Linguistic and Cognitive Engineering [GGM+02] : http ://www.loa.istc.cnr.it/DOLCE.html ?
47. http ://lemon-model.net/ en lien avec le groupe W3C Ontolex : http ://w3.org/community/ontolex.
48. http ://okfn.org/

ces connaissances, qui pourrait être adapté pour Prolmf. D'autre part, nous verrons dans la dernière partie de ce document qu'il est aussi possible d'installer un système de type OBDA (Ontology Based Data Access) qui permettrait d'interroger Prolexbase via une ontologie en laissant les données dans la base relationnelle [49].

Notre étude de cas sur la conception XML révèle bien les difficultés d'une part, mais surtout les convergences des principes et méthodes mis en oeuvre pour parvenir à cette interopérabilité généralisée, au coeur du web, qui permettra de construire nos connaissances et nos outils de connaissance en nous appuyant sur celles et ceux des autres, et vice-versa. Pour le web, on assiste à la montée en puissance du niveau sémantique, avec pour socle l'ontologie comme *représentation consensuelle* d'un domaine de connaissance. En parallèle, et avec le même principe fondamental de *recherche de consensus*, la marche vers la normalisation des ressources linguistiques progresse. Les liens entre ontologies et normes sont évidents et des travaux comme [ZHT12], qui proposent une structuration sémantique du registre de catégories de données ISOCat, montrent que l'exploitation de leurs complémentarités ne fait que commencer.

Ni les ontologies, ni la normalisation ne sont des nouveautés dans le domaine scientifique, loin s'en faut. Comme c'est le web qui suscite leur retour en force au devant de la scène du traitement de l'information, les challenges résident dans les caractéristiques même de ce cadre : large échelle (beaucoup de participants potentiels), dynamicité (on collabore un temps, on arrête, on reprend...), contributions à petite échelle (ontologies "légères" versus "modèles du monde"), inter-relations, incohérence et incomplétude inhérentes, représentations en constante *évolution*. Le chapitre suivant présente nos premières contributions dans ce cadre.

49. Toutefois cette option ne mettrait pas Prolexbase "dans" le web au sens des données liées, c'est-à-dire qu'il ne serait pas possible d'accéder aux données via une URI directement déréferençable.

Chapitre 3

Intégration sémantique de données

Si XML est le format de l'interopérabilité syntaxique, l'interopérabilité sémantique est réalisée par la mise au point de standards et d'ontologies qui, à des niveaux opérationnels différents, décrivent le sens des données en s'appuyant sur un consensus. De telles descriptions permettent une gestion des données plus simple pour les utilisateurs, plus évolutive et interopérable pour les applications informatiques, plus productrices de sens [Bac07]. Il en est ainsi pour les données du web, qui peuvent être décrites par des ontologies. Je considère ici le concept d'ontologie du point de vue du web sémantique, plus restrictif que celui de l'ingénierie des connaissances : les ontologies y sont conçues pour favoriser *l'interopérabilité sémantique* entre applications. Notre contribution dans ce cadre, à travers la thèse de Cheikh Niang, a pour but d'intégrer aisément des ontologies légères représentant des contenus à partager de façon souple et évolutive [NBL10, NBLS11b, NBLS11a, NBLS12, NBLS13, NBSL13].

Je rappelle d'abord le cadre théorique dans lequel s'inscrit cette contribution : la logique de description $DL\text{-}Lite_{\mathcal{A}}$. Ce cadre de représentation de connaissances permet d'exprimer simplement des modèles conceptuels sous la forme d'ontologies [BB03, BCG05, CGL+09, QACT12], qui servent alors d'interfaces entre les utilisateurs et les bases de données selon le principe OBDA (Ontology Based Data Access) [CLL+06, CGL+08, PLC+08, CGL+11]. Ce principe et ses outils déjà développées et encore en cours de développement sont en lien avec le web sémantique (le profil OWL2-QL en est issu) mais pas seulement : ils sont d'un grand intérêt pour des systèmes plus délimités, ils ont déjà été utilisés avec succès dans des systèmes d'information d'entreprises [CGL+11]. Leurs avantages m'ont conduit à diriger vers eux les recherches de Cheikh Niang. Après les avoir présentés je décris donc la contribution : la construction quasi-automatique d'un médiateur sémantique souple et évolutif. Les sources restent indépendantes et sont interrogées de façon transparente pour l'utilisateur lorsqu'une requête est posée au niveau du médiateur. La réponse élaborée est donc toujours à jour par rapport aux données des sources. L'ajout et le retrait d'une source dans ce système d'intégration ont un coût limité tant en temps de calcul qu'en temps d'expertise humaine. Cette solution est *générique*, elle repose sur l'exploitation d'une ontologie de référence du domaine dans lequel la médiation sémantique est souhaitée. Je présente succinctement ensuite sa mise en œuvre dans le cadre du projet Personae, qui représente un contexte d'utilisation typique des principes proposés.

3.1 Des systèmes OBDA : connaissances et données

Notre proposition s'inscrit dans le cadre des systèmes dits OBDA (Ontology-Based Data Access), dans lesquels des ontologies servent d'interfaces pour accéder à des données, comme l'illustre la figure 3.1(a). Plus précisément cette interface est constituée de la partie conceptuelle, ou intentionnelle, de l'ontologie, les instances se trouvant dans la base de données. En effet les ontologies des systèmes OBDA sont représentées par des logiques de description [BCM+03], comme je le rappelle dans la suite la partie conceptuelle est donc une TBox et la partie instances une ABox, laquelle est représentée par un couple $(\mathcal{M}, \mathcal{D})$ où $\mathcal{D}$ est la base de données et $\mathcal{M}$ est un mapping qui permet de générer les instances de la TBox à partir des données stockées dans $\mathcal{D}$. Les logiques de description sont également un des modèles théoriques pour les ontologies du web sémantique. Nous proposons un système d'intégration de systèmes OBDA plus ouvert donc plus adapté au cadre du web sémantique, dont les principes sont schématisés dans la figure 3.1(b). Je présente dans ce qui suit le cadre formel de cette proposition.

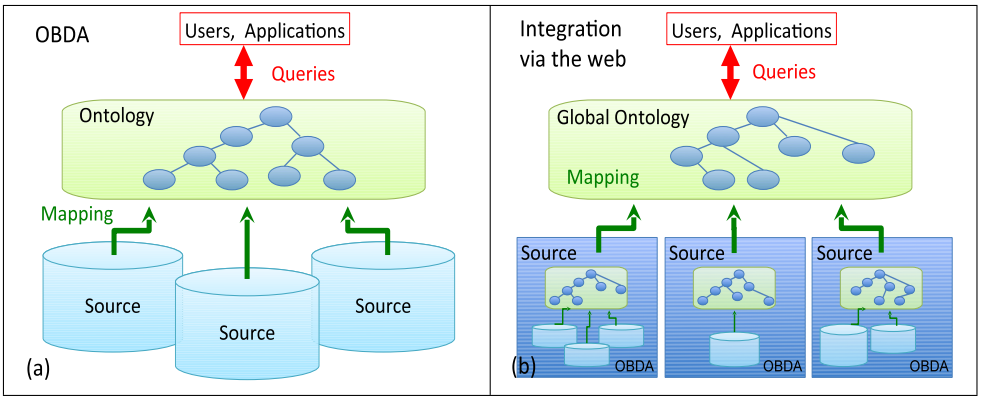


FIGURE 3.1 – OBDA et Web OBDA.

En logique de description, la description conceptuelle des connaissances est appelée une *TBox* et l'ensemble des données qui lui correspondent est appelé une *ABox*. Une TBox $\mathcal{T}$ modélise de façon intentionnelle un domaine d'intérêt en termes (i) de concepts, dénotant un ensemble d'objets, et (ii) de rôles, dénotant des relations binaires entre les objets. Les concepts étant des relations unaires et les rôles des relations binaires, $\mathcal{T}$ est définie (i) par sa "terminologie" (le 'T' de TBox) ou *signature* $sig(\mathcal{T})$, qui est l'union de l'ensemble des concepts atomiques [1] et de l'ensemble des rôles atomiques, et (ii) par un *ensemble de contraintes* portant sur les concepts ou les rôles, contraintes d'inclusion, de disjonction ou encore contraintes fonctionnelles. Une ABox $\mathcal{A}$ est la partie extentionnelle de l'ontologie, un ensemble de faits ou "assertions" (le 'A' de ABox), ou instances des éléments de la TBox correspondante. Ainsi, l'ontologie est formée du couple $\langle \mathcal{T}, \mathcal{A} \rangle$.

Ces ontologies peuvent être utiles à tous les stades du cycle de vie des systèmes d'information, c'est-à-dire déjà lors de la conception, car le niveau intentionnel correspond aux diagrammes de type entités-associations, UML, ou réseaux sémantiques et peut fournir le

1. La notion de concept et de rôle *atomiques* est expliquée au début de la section 3.1.1.

support de ses capacités d'inférence à ce stade [BCG05, QACT12]. Pour l'interrogation, le schéma de l'ontologie peut servir d'interface entre l'utilisateur (ou l'application) et les données, en permettant là aussi un raisonnement sur les connaissances qui enrichit les requêtes et peut également servir de système d'intégration de données [CLL+06, CGL+08, PLC+08, CGL+11] comme le montre la figure 3.1. Les avantages qu'il y a à utiliser de cette manière des schémas d'ontologies définis en logique de description sont donc de combiner l'intuitivité (pour les humains) de la modélisation par diagrammes, avec la calculabilité (raisonnement automatique) de la logique.

En effet, les logiques de description sont des *fragments de la logique du premier ordre* [AvE82, BCM+03], dont l'expressivité est limitée pour permettre des possibilités de raisonnement automatique de complexité raisonnable. Précisément, les éléments de la famille des logiques de description *DL-Lite* ont pour particularité d'être *réductibles à la logique du premier ordre*, ce qui signifie qu'une requête impliquant les données d'une ontologie $\langle \mathcal{T}, \mathcal{A} \rangle$ peut être traitée en deux étapes, d'abord un raisonnement qui n'utilise que $\mathcal{T}$ pour produire une requête en logique du premier ordre et ensuite l'évaluation de cette requête sur la partie $\mathcal{A}$ comme sur une base de données relationnelles (avec l'avantage des stratégies d'optimisation bien établies dans ce cadre). L'ensemble des correspondances entre (i) les notations en logique de description DL-Lite, (ii) les notations en logique du premier ordre et (iii) les notations en relationnel sont présentées précisément dans [GR11].

Les différentes logiques de description se distinguent par les contraintes qu'elles proposent pour spécifier les concepts et les rôles. C'est ainsi que les langages de description des ontologies du web sémantique OWL1 [MvH04] et OWL2 [HKP+09] sont déclinés en parties qui correspondent à différents types de logique de description [2]. OWL2 définit ainsi trois profils ciblés pour des usages précis afin d'offrir une meilleure efficacité dans ces contextes, OWL2 QL, OWL2 EL, OWL2 RL. Le profil OWL2 QL [MGH+09] dérive de la famille de logiques de description DL-Lite. Il permet l'expression des caractéristiques essentielles des diagrammes de classes UML et des schémas Entités-Associations et il est destiné aux applications qui utilisent de gros volumes de données et pour lesquelles la réponse à des requêtes est la principale tâche de raisonnement. L'idée est, dans ce cas de figure, de confier la gestion des données (la ABox) à un système de gestion de bases de données relationnelles, pour profiter de l'ensemble de ses optimisations, ce qui revient à un système OBDA.

Un système OBDA intègre donc des bases de données relationnelles. Cela suppose (i) de concevoir le schéma de l'ontologie (la TBox) et (ii) de définir des mappings entre ce schéma et les données relationnelles, ou vues, qui représentent la ABox. Ainsi, il reste des challenges à relever comme (i) une aide pour la conception de l'ontologie (schéma), (ii) une aide pour la construction des mappings entre ce schéma et les données relationnelles et (iii) une assistance pour l'évolution du système (schéma, mappings, ajout et retrait de sources de données). Notre contribution n'est pas une réponse directe à ces défis mais une solution de contournement intéressante dans le cadre du web. Comme l'illustre la figure 3.1, en partant de l'hypothèse que des sources ont déjà produit un schéma d'ontologie pour représenter leurs données (pratique sous-tendue par le web sémantique), elle consiste à ajouter un niveau supplémentaire d'intégration : l'intégration d'ontologies légères au sein d'un système de

2. RDFS (sans nœud vide), langage de description de schémas RDF, et donc premier niveau de langage d'ontologie, correspond à un fragment de la logique de description $DL\text{-}Lite_{\mathcal{R}}$, décrite dans la suite de ce document.

médiation. Nous utilisons la logique de description *DL-Lite*$_\mathcal{A}$ [CGL+07, PLC+08] pour formaliser et exploiter ces ontologies. Dans la suite, je rappelle d'abord les définitions de cette logique de description, puis je montre dans un exemple ses liens avec les diagrammes de classes UML, avant d'exposer les principes de l'évaluation de requêtes dans un système OBDA.

3.1.1 Syntaxe et sémantique de la logique de description *DL-Lite*$_\mathcal{A}$

En *DL-Lite*$_\mathcal{A}$ le schéma conceptuel est décrit en termes de concepts, rôles et attributs (valeurs de propriétés de concepts), avec les notations suivantes [CGL+07, GR11] :

$$B ::= A \mid \exists Q \mid \delta(U) \qquad Q ::= P \mid P^- \qquad E ::= \rho(U)$$
$$C ::= \top_C \mid B \mid \neg B \qquad R ::= Q \mid \neg Q \qquad F ::= \top_D \mid T_1 \mid ... \mid T_n$$
$$V ::= U \mid \neg U$$

où A est un *concept atomique* (c'est-à-dire un nom de concept), B un *concept basique*, C un *concept général* et $\top_C$ le *concept universel*. De même, P dénote un *rôle atomique*, Q un *rôle basique*, et R un *rôle général*. E est un *type de valeurs basique* (par exemple le type *string*), F l'*expression d'un type de valeurs* (correspondant aux types de valeurs définis dans RDF), et $\top_D$ le *type de valeurs universel*. Enfin, U est un *attribut atomique* et V *un attribut général*. $\neg B$, $\neg Q$ et $\neg U$ servent à exprimer des contraintes de disjonction.

La sémantique d'une expression *DL-Lite*$_\mathcal{A}$ est donnée en termes d'*interprétations*. Une interprétation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ est formée du domaine d'interprétation $\Delta^\mathcal{I}$ et de la fonction d'interprétation $\cdot^\mathcal{I}$. Le domaine $\Delta^\mathcal{I} = \Delta^\mathcal{I}_O \cup \Delta^\mathcal{I}_V$, est un ensemble non vide où $\Delta^\mathcal{I}_O$ est le *domaine des objets* et $\Delta^\mathcal{I}_V$ est le *domaine des valeurs*. La fonction $\cdot^\mathcal{I}$ associe un sous ensemble de $\Delta^\mathcal{I}$ à chaque concept ou domaine de valeurs, et un sous ensemble de $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ à chaque rôle ou attribut, de telle sorte que les conditions suivantes soient satisfaites :

rôle atomique :	$P^\mathcal{I} \subseteq \Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O$
rôle inverse :	$(P^-)^\mathcal{I} = \{(o, o') \mid (o', o) \in P^\mathcal{I}\}$
négation de rôle :	$(\neg Q)^\mathcal{I} = (\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O) \backslash Q^\mathcal{I}$
concept atomique :	$A^\mathcal{I} \subseteq \Delta^\mathcal{I}_O$
restriction existentielle :	$(\exists Q)^\mathcal{I} = \{o \mid \exists o'.(o, o') \in Q^\mathcal{I}\}$
négation de concept :	$(\neg B)^\mathcal{I} = \Delta^\mathcal{I}_O \backslash B^\mathcal{I}$
domaine d'attribut :	$(\delta(U))^\mathcal{I} = \{o \mid \exists v.(o, v) \in U^\mathcal{I}\}$
concept universel :	$(\top_C)^\mathcal{I} = \Delta^\mathcal{I}_O$
attribut atomique :	$U^\mathcal{I} \subseteq \Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_V$
négation d'attribut :	$(\neg U)^\mathcal{I} = (\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_V) \backslash U^\mathcal{I}$
type universel :	$(\top_D)^\mathcal{I} = \Delta^\mathcal{I}_V$
type de données :	$(T_i)^\mathcal{I} \subseteq \Delta^\mathcal{I}_V$
co-domaine d'attribut :	$(\rho(U))^\mathcal{I} = \{v \mid \exists o.(o, v) \in U^\mathcal{I}\}$
constante objet :	$c^\mathcal{I} \in \Delta^\mathcal{I}_O$
constante valeur :	$val(d) \in \Delta^\mathcal{I}_V$

La ***TBox*** $\mathcal{T}$ décrit la connaissance sur les concepts et les rôles via un ensemble fini d'assertions, ou axiomes. Un axiome en *DL-Lite*$_\mathcal{A}$ est soit une contrainte d'inclusion, soit

une contrainte de disjonction, soit une contrainte fonctionnelle, soit une contrainte d'identification. Une ***contrainte d'inclusion***, ou *inclusion positive* (**PI**), est de la forme :

$B_1 \sqsubseteq B_2$ (inclusion de concepts)
$Q_1 \sqsubseteq Q_2$ (inclusion de rôles)
$U_1 \sqsubseteq U_2$ (inclusion d'attributs)
$\rho(U) \sqsubseteq (T_i)$ (appartenance des valeurs d'un attribut à un type)

Une inclusion de concept (respectivement, rôle, et attribut) exprime le fait que l'ensemble des instances d'un concept basique B_1 (respectivement, rôle basique Q_1, et attribut basique U_1) est inclus dans l'ensemble des instances du concept B_2 (respectivement, rôle Q_2, et attribut U_2), plus général. C'est la relation de subsomption, qui structure le modèle conceptuel en taxonomies, ou hiérarchies.

Une ***contrainte de disjonction***, ou *inclusion négative* (**NI**), est de la forme :

$B_1 \sqsubseteq \neg B_2$ (aucune instance de B_1 ne peut être instance de B_2)
$Q_1 \sqsubseteq \neg Q_2$
$U_1 \sqsubseteq \neg U_2$

Une ***contrainte fonctionnelle*** sur un rôle ou un attribut est de la forme : $(funct\ Q)$ ou $(funct\ U)$ (unicité de ce rôle ou de cet attribut pour chaque instance de leur domaine). Une ***contrainte d'identification*** (d'un concept B) est de la forme $(id\ B\ I_1, \ldots, I_n)$ où chaque I_j est un rôle, un inverse de rôle ou un attribut.

Une ***ABox*** en $DL\text{-}Lite_{\mathcal{A}}$ consiste en un nombre fini d'assertions de faits, pour des concepts, attributs et rôles atomiques de la forme : $A(c)$, $P(c, c')$ et $U(c, d)$, où c et c' sont des constantes objets et d est une constante valeur.

La sémantique d'une TBox $DL\text{-}Lite_{\mathcal{A}}$ est définie en spécifiant quand une interprétation $\mathcal{I}$ satisfait un axiome α, ce qui se note : $\mathcal{I} \models \alpha$. Précisément, il s'agit de vérifier les conditions suivantes :

Contrainte	**Sémantique**
inclusion de concepts	${B_1}^{\mathcal{I}} \subseteq {B_2}^{\mathcal{I}}$
inclusion de rôles	${Q_1}^{\mathcal{I}} \subseteq {Q_2}^{\mathcal{I}}$
inclusion d'attributs	${U_1}^{\mathcal{I}} \subseteq {U_2}^{\mathcal{I}}$
inclusion valeur - type	$\rho(U)^{\mathcal{I}} \subseteq {T_i}^{\mathcal{I}}$
disjonction de concepts	${B_1}^{\mathcal{I}} \subseteq (\neg B_2)^{\mathcal{I}}$
disjonction de rôles	${Q_1}^{\mathcal{I}} \subseteq (\neg Q_2)^{\mathcal{I}}$
disjonction d'attributs	${U_1}^{\mathcal{I}} \subseteq (\neg U_2)^{\mathcal{I}}$
contrainte fonctionnelle sur rôle $(funct\ Q)$	$\forall o, o_1, o_2.(o, o_1) \in Q^{\mathcal{I}} \wedge (o, o_2) \in Q^{\mathcal{I}} \rightarrow o_1 = o_2$
contrainte fonctionnelle sur attribut $(funct\ U)$	$\forall o, v, v'.(o, v) \in U^{\mathcal{I}} \wedge (o, v') \in U^{\mathcal{I}} \rightarrow v = v'$
contrainte d'identification $(id\ B\ I_1, \ldots, I_n)$	$\forall o_1, o_2 \in B^{\mathcal{I}}$ et pour tous objets ou valeurs $u_1, \ldots, u_n$ on a : $(o_1, u_j) \in I_j^{\mathcal{I}} \wedge (o_2, u_j) \in I_j^{\mathcal{I}}$ pour $j \in \{1, \ldots, n\} \rightarrow o_1 = o_2$ [3]

Table 3.1 – Sémantique d'une TBox

De même, la sémantique d'une ABox *DL-Lite*$_\mathcal{A}$ est définie en spécifiant quand une interprétation $\mathcal{I}$ satisfait les assertions d'appartenance suivantes :

$A(c) : c^\mathcal{I} \in A^\mathcal{I}$
$P(c, c') : ({c_1}^\mathcal{I}, {c_2}^\mathcal{I}) \in P^\mathcal{I}$
$U(c, d) : (c^\mathcal{I}, val(d)) \in U^\mathcal{I}$

La sémantique pour les individus suit les hypothèses de *nom unique* (deux individus différents ont deux interprétations différentes) et de *nom standard* (quelle que soit l'interprétation $\mathcal{I}$, $c^\mathcal{I} = c$).

Une interprétation $\mathcal{I}$ est un modèle d'une TBox $\mathcal{T}$, ou $\mathcal{I}$ satisfait $\mathcal{T}$, ce qui se note $\mathcal{I} \models \mathcal{T}$, si et seulement si $\mathcal{I}$ satisfait tous les axiomes de $\mathcal{T}$. Une interprétation $\mathcal{I}$ est un modèle d'une ABox $\mathcal{A}$ si et seulement si elle satisfait toutes les assertions de $\mathcal{A}$. C'est un modèle de l'ontologie $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ si et seulement si elle est un modèle pour $\mathcal{T}$ et pour $\mathcal{A}$. Une ontologie $\mathcal{O}$ (respectivement une TBox $\mathcal{T}$) implique logiquement une assertion α, ce qui se note $\mathcal{O} \models \alpha$ (respectivement $\mathcal{T} \models \alpha$), si α est satisfaite par tous les modèles de $\mathcal{O}$ (respectivement $\mathcal{T}$). Il y a trois formes de raisonnement sur une ontologie $\mathcal{O}$ en *DL-Lite*$_\mathcal{A}$: (1) la *satisfiabilité* ou vérifier si $\mathcal{O}$ admet au moins un modèle, (2) le *test d'appartenance* d'instances de concept ou de rôle ou vérifier si $\mathcal{O} \models C(c)$ ou $\mathcal{O} \models R(c_1, c_2)$ et (3) *l'évaluation de requête.*

Pour limiter la complexité de ces raisonnements, des restrictions sont imposées sur les assertions d'une TBox en *DL-Lite*$_\mathcal{A}$: *aucun rôle ou attribut fonctionnel ou identifiant ne peut être spécialisé (c'est-à-dire apparaître du côté droit d'une inclusion).* En d'autres termes, s'il y a une contrainte de fonctionnalité sur P ou P^- (respectivement U), ou si P ou P^- (respt. U) intervient dans une contrainte d'identification, alors il ne peut y avoir d'assertion $Q \sqsubseteq P$ ni $Q \sqsubseteq P^-$ (respt. $U' \sqsubseteq U$) dans $\mathcal{T}$. Dans ces conditions, les tâches de raisonnement citées précédemment sont polynomiales en temps (appartiennent à la classe de problèmes $PTIME$) sur la taille de la TBox et constantes en temps en utilisant un nombre polynomial de processeurs (classe AC^0) sur la taille de la ABox, ce qui est la complexité de l'évaluation de requêtes dans le modèle relationnel. La *DL-Lite*$_\mathcal{A}$ est la plus expressive des logiques de descriptions qui conserve ces bonnes propriétés [CGL+07].

Les deux autres membres de la famille DL-Lite sont des sous-languages de la *DL-Lite*$_\mathcal{A}$: *DL-Lite*$_\mathcal{F}$ permet les contraintes de fonctionnalité mais pas les inclusions de rôles et *DL-Lite*$_\mathcal{R}$ permet les inclusions de rôles mais pas les contraintes de fonctionnalité. Aucun des deux ne traite les valeurs (les attributs et leurs types). Dans notre système d'intégration actuel, nous n'utilisons pas les contraintes fonctionnelles ni les contraintes d'identification, mais nous considérons les attributs et leurs valeurs.

3.1.2 TBox *DL-Lite*$_\mathcal{A}$ et diagramme de classes UML

Afin d'une part de donner une intuition sur le sens des expressions *DL-Lite*$_\mathcal{A}$, et d'autre part de justifier le principe OBDA (un schéma d'ontologie pour accéder aux données), considérons sur un exemple comment le niveau intentionnel d'une ontologie *DL-Lite*$_\mathcal{A}$, c'est-à-dire sa TBox, correspond à un diagramme de classes UML, limité à des classes, des attributs, des associations et la relation d'héritage. La figure 3.2 représente une partie d'un diagramme de classes UML, représentant la base de données des chantres dans le projet

Personae dont il sera question à la fin de ce chapitre.

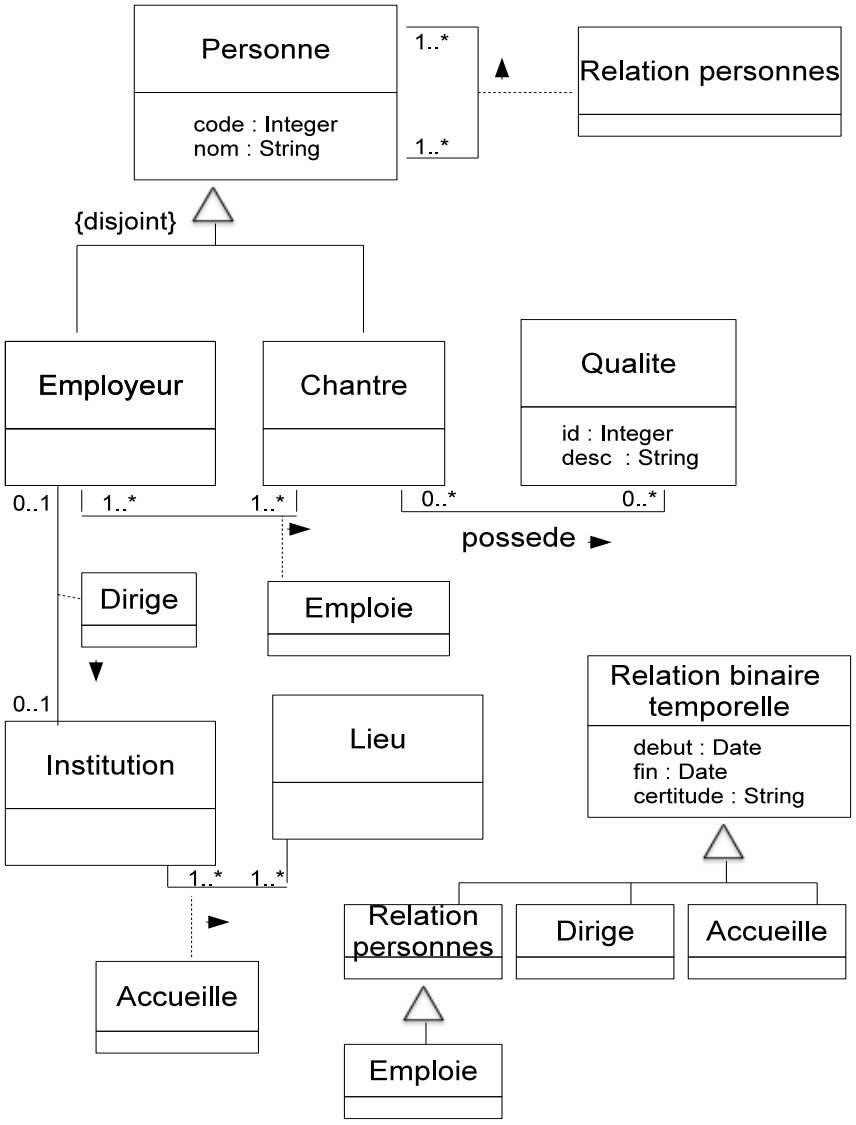


Figure 3.2 – Exemple de diagramme de classes.

D'après [BB03, BCG05] et surtout [CGL+09], ce schéma conceptuel peut s'exprimer en $DL\text{-}Lite_{\mathcal{A}}$ de la manière suivante :

- Une classe A en UML est un concept A en $DL\text{-}Lite_{\mathcal{A}}$. Par exemple, à partir du diagramme en figure 3.2, nous aurons les concepts *Personne*, *Employeur*, etc.
- Un attribut a de type T d'une classe A en UML est exprimé en $DL\text{-}Lite_{\mathcal{A}}$ par : (i) $\delta(a_A) \sqsubseteq A$ (une instance du domaine de l'attribut a_A est une instance de A) et (ii)

$\rho(a_A) \sqsubseteq (xsd : T)$ (l'attribut a_A est du type T défini dans la spécification du langage XML Schema - *xsd* :). Ainsi dans l'exemple, l'attribut *code* de la classe $Personne$ est représenté par (i) $\delta(code_{Personne}) \sqsubseteq Personne$ et (ii) $\rho(code_{Personne}) \sqsubseteq (xsd : int)$. "Contextualiser" le nom de l'attribut par le nom de la classe est nécessaire car en UML deux classes distinctes peuvent avoir chacune un attribut de même nom.

- Pour exprimer que l'attribut a de la classe A est obligatoire on écrit : $A \sqsubseteq \delta(a_A)$ (toute instance de A a l'attribut a_A - ou "est dans le *domaine* de a_A" -), par exemple $Personne \sqsubseteq \delta(code_{Personne})$; pour exprimer que cet attribut est unique on écrit $(funct\ a_A)$, comme par exemple $(funct\ code_{Personne})$.
- Les relations d'héritage de UML sont représentées en *DL-Lite*$_\mathcal{A}$ par des inclusions positives. Ainsi dans la figure 3.2 nous avons par exemple $Chantre \sqsubseteq Personne$.
- Le fait que $Employeur$ et $Chantre$ soient deux classes disjointes s'exprime en *DL-Lite*$_\mathcal{A}$ par une inclusion négative : $Employeur \sqsubseteq \neg Chantre$.
- Une association P de la classe A vers la classe A' est représentée en *DL-Lite*$_\mathcal{A}$ par deux inclusions, qui définissent le domaine et (respectivement) le co-domaine du rôle correspondant : $\exists P \sqsubseteq A$ et (respectivement) $\exists P^- \sqsubseteq B$. Par exemple pour l'association *possede*, on a $\exists possede \sqsubseteq Chantre$ et $\exists possede^- \sqsubseteq Qualite$.
- De même que pour les attributs, pour représenter les cardinalités (min, max) du nombre d'instances de A' impliquées dans l'association P pour chaque instance de A, on écrit $A \sqsubseteq \exists P$ si $min = 1$ et $(funct\ P)$ si $max = 1$. Réciproquement, on écrira $A' \sqsubseteq \exists P^-$ si pour chaque instance de A' il doit y avoir au moins une instance de A impliquée dans l'association P et $(funct\ P^-)$ si pour chaque instance de A' il doit y avoir au plus une instance de A impliquée dans l'association P.
- Une association A entre plus de deux classes ou une association pour laquelle est définie une classe d'association (comme entre $Employeur$ et $Institution$ de la figure 3.2, avec la classe d'association *dirige*) se représente en appliquant une *réification* : l'association devient un concept A et on définit autant de rôles qu'il y a de classes associées, entre les concepts correspondant à ces classes et le concept A correspondant à l'association. Pour deux classes C_1 et C_2 reliées par une classe d'association A on a les rôles $R_{A,1}$ et $R_{A,2}$ et pour préciser que chaque instance de A participe exactement une fois à la fois au rôle $R_{A,1}$ et au rôle $R_{A,2}$ on écrit : $A \sqsubseteq R_{A,1}$, $(funct\ R_{A,1})$, $A \sqsubseteq R_{A,2}$ et $(funct\ R_{A,2})$. Pour préciser que A relie les classes C_1 et C_2 on écrit : $\exists R_{A,1} \sqsubseteq A$, $\exists R^-_{A,1} \sqsubseteq C_1$, $\exists R_{A,2} \sqsubseteq A$, $\exists R^-_{A,2} \sqsubseteq C_2$. Enfin on écrit $(id\ A\ R_{A,1}, R_{A,2})$ pour préciser que chaque instance de A représente un couple distinct dans $C_1 \times C_2$. Dans notre exemple, pour l'association *dirige* on doit créer le concept $Dirige$ et on doit décrire de la façon qui vient d'être détaillée deux rôles : l'un entre les concepts $Employeur$ et $Dirige$ d'une part, et l'autre entre $Dirige$ et $Institution$ d'autre part.

Inversement, on peut représenter en UML des ontologies décrites en *DL-Lite*$_\mathcal{A}$. La transformation, dans un sens comme dans l'autre, n'est pas sans perte d'information : par exemple une partie seulement des contraintes de cardinalité s'expriment en *DL-Lite*$_\mathcal{A}$ avec les inclusions, les contraintes fonctionnelles et les contraintes d'identité, et sans l'union de concepts, qui n'existe pas en *DL-Lite*$_\mathcal{A}$, on ne peut pas exprimer un fait comme "la classe Personne est exactement l'union des classes Employeur et Chantre". Il n'en reste pas moins que ce qui est exprimable dans les deux à la fois forme un noyau dont l'expressivité est suffisante pour un grand nombre de cas, surtout comme schéma d'interrogation, ce qui est

l'objet des systèmes OBDA.

3.1.3 Requêtes sur une ontologie $DL\text{-}Lite_{\mathcal{A}}$: les principes OBDA

Le principe d'un système OBDA est que la ABox de l'ontologie ne soit pas gérée directement par le raisonneur (en mémoire vive ou secondaire) mais soit laissée ou mise sous la forme d'une base de données relationnelle. C'est utile lorsque les données sont volumineuses, ou qu'elles appartiennent à un tiers ou encore pour intégrer plusieurs bases de données (cf. figure 3.1). Interroger une ontologie $DL\text{-}Lite_{\mathcal{A}}$ avec les principes OBDA repose donc en partie sur le SGBD relationnel. Dans un système relationnel, la requête SQL est écrite selon le schéma logique de la base et répondre à une requête revient à l'évaluer sur les données de la base en considérant celles-ci complètes (*hypothèse du monde clos*). Par contre, le niveau intensionnel d'une ontologie décrit un ensemble de contraintes et les instances peuvent être incomplètes ou incohérentes vis-à-vis de ces contraintes : interroger à ce niveau implique des mécanismes d'inférences logiques [CGL+13].

Interroger une ontologie $DL\text{-}Lite_{\mathcal{A}}$ $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ consiste à définir une expression au niveau intensionnel ($\mathcal{T}$) qui dénote un ensemble de n-uplets d'instances qui satisfont une contrainte. L'évaluation de cette expression est l'une des trois formes de raisonnement possibles sur $\mathcal{O}$, évoquées en section 3.1.1. La définition de cette expression fait intervenir les concepts et les rôles définis dans $\mathcal{T}$ et les constantes de $\mathcal{A}$, elle doit permettre d'interroger l'ontologie un peu comme on interrogerait une base de données. Pour cela, la forme choisie pour les requêtes dans un système OBDA est l'union de requêtes conjonctives [CGL+07] et on utilise la notation *Datalog* [AHV95] suivante pour une requête :

$$q(\bar{x}) \leftarrow conj(\bar{x}, \bar{y}),$$

où $q(\bar{x})$ est appelée *tête* de la requête et $conj(\bar{x}, \bar{y})$ son *corps*. Dans cette requête, $\bar{x}$ est un tuple de variables, les *variables libres* pour lesquelles on recherche une valuation, et $\bar{y}$ est un tuple de *variables existentielles*. Le corps $conj(\bar{x}, \bar{y})$ est une conjonction d'atomes dont les variables sont uniquement des variables de $\bar{x}$ et de $\bar{y}$, et dont les prédicats sont des concepts atomiques ($A(x)$), des rôles atomiques ($P(x, y)$) ou des attributs atomiques ($U_C(x, y)$) de la TBox $\mathcal{T}$. L'*arité* d'une requête est le nombre de ses variables libres $\bar{x}$, et lorsque $\bar{x}$ est le tuple $\langle\rangle$ d'arité 0, alors q est appelée une *requête booléenne*. *Une union de requêtes conjonctives* q est un ensemble de requêtes conjonctives qui ont toutes la même tête, ce qui, avec la notation *Datalog*, s'écrit de la manière suivante :

$$q(\bar{x}) \leftarrow conj_1(\bar{x}, \bar{y}_1)$$
$$\ldots\ldots\ldots$$
$$q(x) \leftarrow conj_n(\bar{x}, \bar{y}_n)$$

Voici par exemple une requête sur une ontologie dont la TBox correspondrait au schéma conceptuel de la figure 3.2 (elle interroge les qualités possédées par au moins un chantre) :

$q(x_1, x_2) \leftarrow \exists(y, q).Chantre(y), possede(y, q), id(q, x_1), desc(q, x_2)$
ou, en notation Datalog :
$q(x_1, x_2) \leftarrow Chantre(y), possede(y, q), id(q, x_1), desc(q, x_2)$

Étant données une ontologie $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, une interprétation $\mathcal{I}$ de $\mathcal{O}$ et une requête $q(\bar{x})$, la *réponse* à $q(\bar{x})$ sur $\mathcal{I}$, notée $ans(q, \mathcal{O}, \mathcal{I}) = q^{\mathcal{I}}$, est l'ensemble de n-uplets $\bar{c}$ de constantes de

$\mathcal{A}$ pour lesquels $conj(\bar{c}, \bar{y})$ est vrai dans $\mathcal{I}$. La *réponse certaine* à $q(\bar{x})$, notée $cert(q, \mathcal{O})$, est l'ensemble de n-uplets $\bar{c}$ de constantes de $\mathcal{A}$ qui appartiennent à $q^{\mathcal{I}}$ quelle que soit $\mathcal{I}$. Lorsque q est une requête booléenne, $cert(q, \mathcal{O})$ est définie par *true* si $[\exists \bar{y}\ conj(\emptyset, \bar{y})]^{\mathcal{I}} = true$, et *false* sinon.

Concernant une requête booléenne, la notion *d'interprétation canonique* sera utile : l'interprétation canonique d'une requête booléenne $q() \leftarrow conj(\bar{y})$ est $\mathcal{I}_{\text{II}} = (\Delta^{\mathcal{I}_{\text{II}}}, \cdot^{\mathcal{I}_{\text{II}}})$ où (i) $\Delta^{\mathcal{I}_{\text{II}}}$ est l'ensemble des variables et des constantes dans $conj(\bar{y})$ et (ii) $\cdot^{\mathcal{I}_{\text{II}}}$ est telle que $c^{\mathcal{I}_{\text{II}}} = c$ pour toutes les constantes dans $conj(\bar{y})$ et $(t_1, \ldots, t_k) \in P^{\mathcal{I}_{\text{II}}}$ si et seulement si l'atome $P(t_1, \ldots, t_k)$ apparaît dans $conj(\bar{y})$. Par exemple pour la requête :

$q() \leftarrow Personne(y), code(y, a),$

le domaine de l'interprétation canonique est $\Delta^{\mathcal{I}_{\text{II}}} = \{y, a\}$ et on a : $Personne^{\mathcal{I}_{\text{II}}} = \{(y)\}$, $code^{\mathcal{I}_{\text{II}}} = \{(y, a)\}$ et $a^{\mathcal{I}_{\text{II}}} = a$. Par extension, on peut également parler d'interprétation canonique pour une requête quelconque en "gelant" ses variables libres (c'est-à-dire en en faisant des constantes), ce qui a été la base d'algorithmes de tests d'inclusion de requêtes [AHV95].

La notion d'interprétation canonique peut donc être utilisée pour vérifier la satisfiabilité potentielle d'une requête formulée sur la TBox. Plus généralement, dans une ontologie en $DL\text{-}Lite_{\mathcal{A}}$, les inclusions négatives et les contraintes fonctionnelles introduisent des négations, ce qui implique que l'ontologie puisse être incohérente. La cohérence (ou satisfiabilité) d'une ontologie $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ contenant des inclusions négatives et des contraintes fonctionnelles peut être vérifiée en construisant une requête de la logique du premier ordre q_{unsat} telle que $cert(q_{unsat}, \mathcal{O}) = \{true\}$ si et seulement si l'ontologie $\langle \mathcal{T}, \mathcal{A} \rangle$ est incohérente pour toute ABox $\mathcal{A}$ [CGL+07]. L'algorithme pour construire la requête q_{unsat} [CGL+07] revient à calculer la fermeture transitive de l'ensemble des inclusions négatives et des contraintes fonctionnelles, à transformer chaque élément de cette fermeture transitive en une requête conjonctive booléenne et à rassembler ces dernières dans q_{unsat} (union de requêtes conjonctives). Cet algorithme a une complexité polynomiale sur la taille de la TBox. Dans notre exemple de la figure 3.2, on aurait la requête conjonctive booléenne $unsat() \leftarrow Employeur(x), Chantre(x)$ qui, lorsqu'elle est évaluée à *true* démontre une incohérence dans les données.

Si l'ontologie est cohérente alors il est sensé de l'interroger. En $DL\text{-}Lite_{\mathcal{A}}$ le calcul des réponses certaines à une requête est réalisé en deux étapes : une inférence sur la TBox seulement, puis une évaluation sur la ABox uniquement. Dans un système OBDA, la ABox est une base de données, il faut donc un *mapping* entre la base de données et la TBox, qui comprenne un mécanisme pour obtenir à partir de la base de données les instances de la TBox. Le principe, décrit dans [CLL+06, CGL+07], est de produire les instances à partir des identifiants dans la base. Par exemple pour la base des chantres (dont un extrait est en figure 3.2), les instances du concept $Chantre$ forment l'ensemble $\{Chantre(pers(code_i))\}$ pour $1 \leq i \leq n$, où $pers$ est un terme choisi par la personne qui définit le mapping et les $code_i$ sont les n valeurs de l'attribut $code$ extraites de la table $Chantre$.

Formellement, un système OBDA est donc un triplet $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$, où $\mathcal{T}$ est une TBox, $\mathcal{M}$ est un ensemble d'assertions de mapping entre $\mathcal{T}$ et $\mathcal{D}$, enfin $\mathcal{D}$ est la base de données. Les assertions de mapping sont de la forme $\Phi(\bar{x}) \rightsquigarrow \Psi(\bar{x})$, où Φ est une requête SQL d'arité n sur $\mathcal{D}$ et Ψ est une requête conjonctive sur $\mathcal{T}$ d'arité n' dont les variables libres

sont dans $\bar{x}$. Φ est le corps de l'assertion de mapping et Ψ en est la tête. Intuitivement, une interprétation $\mathcal{I}$ satisfait le mapping $\Phi \rightsquigarrow \Psi$ par rapport à la base $\mathcal{D}$ si tous les faits obtenus en évaluant Φ sur $\mathcal{D}$ et en propageant les réponses dans Ψ sont vérifiés dans $\mathcal{I}$. Le calcul des réponses certaines à une requête q posée sur un système OBDA passe alors par les étapes listées dans la définition suivante [CGL+08] :

Définition 3.1.1 - ***étapes de calcul des réponses à une requête posée sur un système OBDA :***

1. Ré-écriture de q en q_{re} telle que $cert(q, \langle \mathcal{T}, \mathcal{A} \rangle) = cert(q_{re}, \mathcal{A})$. Cette étape est réalisée par l'algorithme $PerfectRef$ [CGL+07, PLC+08]. Le principe est d'appliquer les inclusions positives de la TBox aux atomes de la requête, en chaînage arrière, pour que toute les connaissances exprimées dans la TBox soient bien utilisées dans le calcul des réponses, comme le montre intuitivement l'exemple 3.1.1. Cet algorithme est dans la classe de problèmes PTIME.
2. Dépliage [4] de q_{re} qui utilise les assertions de mapping dans $\mathcal{M}$ pour ré-écrire chaque atome de q_{re}, exprimé sur $\mathcal{T}$, en une requête sur $\mathcal{D}$. La requête obtenue, q_{de}, est l'union de toutes les ré-écritures possibles pour chacun des atomes, soit jusqu'à 2^n ré-écritures si n est le nombre d'atomes de la requête q_{re}.
3. Evaluation de q_{de} sur $\mathcal{D}$, déléguée au système de gestion de base de données.

Exemple 3.1.1 ***Illustration de l'algorithme PerfectRef :***
Soit la TBox constituée des 3 assertions suivantes :
(1) $Chantres \sqsubseteq Personne$ *(toute instance de la classe Chantre est instance de la classe Personne),*
(2) $Personne \sqsubseteq \exists travaillePour$ *(pour toute instance x de Personne il existe une instance y telle que* $travaillePour(x, y)$*) et*
(3) $\exists travaillePour^{-} \sqsubseteq Institution$ *(pour toute instance x, s'il existe une instance* $travaillePour(y, x)$ *alors x est une instance de Institution).*

Soit la ABox constituée des assertions suivantes : $travaillePour$*(HervéAlbi, ChapelleDucaleCourDeSavoie),* $travaillePour$*(JeanDuHamel, ChapelleRoyaleCourDeFrance),* $Chantre$*(JeanGossuin) et* $Chantre$*(JeanLeMaçon).*

Prenons la requête suivante :
$q(x) \leftarrow travaillePour(x, y), Institution(y)$
sans les connaissances de la TBox il n'y a pas d'instance pour y répondre.

En utilisant toute la connaissance codée dans la TBox, l'algorithme PerfectRef transforme la requête q en l'union de toutes les requêtes suivantes :
$q(x) \leftarrow travaillePour(x, y), Institution(y)$
$q(x) \leftarrow travaillePour(x, y), travaillePour(\_, y)$ *(– assertion (3) de la TBox –)*
$q(x) \leftarrow travaillePour(x, \_)$ *(– unification de variables (x) et simplification –)*
$q(x) \leftarrow Personne(x)$ *(– assertion (2) de la TBox –)*
$q(x) \leftarrow Chantre(x)$ *(– assertion (1) de la TBox –)*

4. Du fait que le mapping défini est de type GAV, voir section 3.2.

En évaluant ces ré-écritures sur la ABox on obtient alors comme réponse l'ensemble {*HervéAlbi, JeanDuHamel, JeanGossuin et JeanLeMaçon*}.

L'exemple précédent illustre la première étape du calcul des réponses certaines à une requête q dans un système OBDA $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$ (définition 3.1.1), qui est une *ré-écriture de la requête* par inférences, qui se calcule sur la TBox $\mathcal{T}$ uniquement. La deuxième étape est un calcul sur les mappings $\mathcal{M}$ uniquement, qui permet d'obtenir les requêtes SQL à exécuter sur les sources. Elle est suivie de *l'évaluation* (recherche des valeurs), qui se fait sur la base de données $\mathcal{D}$ uniquement. De ce fait, la complexité du calcul des réponses se décline en fonction des données ($\langle \mathcal{M}, \mathcal{D} \rangle$), en fonction de la requête (q) et en fonction du schéma ($\mathcal{T}$). Il est démontré que la complexité de la ré-écriture est *polynomiale* sur la taille de la TBox $\mathcal{T}$ et *exponentielle* sur la taille de la requête q (du fait de l'étape 2). La complexité de l'évaluation est *logarithmique* sur la taille de la ABox, représentée par $\mathcal{M}$ et $\mathcal{D}$.

Ces principes généraux ont été régulièrement raffinés par leurs auteurs pour améliorer l'efficacité des systèmes OBDA dans les situations réelles dans lesquelles ils sont utilisés, à savoir des systèmes d'information extrêmement complexes de grandes entreprises et administrations italiennes. Ainsi le système MASTRO [5] [CGL+11, GLL+12] implémente une version dans laquelle toute ré-écriture incluse dans une autre est évitée grâce à une analyse intermédiaire menée sur une version *Datalog* du système [RA10, PLL+13]. Par ailleurs, ses auteurs ont encore optimisé son fonctionnement en introduisant dans l'étape 2 de la définition 3.1.1 des ré-écritures induites d'assertions d'inclusion entre les mappings dans $\mathcal{M}$: l'algorithme de ré-écriture *PerfectMap* [PLL+13] s'applique sur le résultat de l'étape 1 dans la définition 3.1.1 (utilisant une version optimisée de PerfectRef), c'est à dire une union de requêtes conjonctives sur $\mathcal{T}$, et génère des requêtes SQL minimales [6] en utilisant des mappings ré-écrits selon les inclusions connues, lesquelles peuvent être découvertes automatiquement par des raisonneurs.

Ontop [7] [RMC12] est un autre système OBDA qui implémente d'autres types d'optimisation à partir de méta-informations sur la base relationnelle interrogée. Nous utilisons ce dernier dans le projet Personae décrit en section 3.3, pour associer à chaque source (base de données relationnelle) une ontologie légère afin d'interroger la source via cette ontologie, sans rien modifier au fonctionnement de sa base pour les autres applications qui l'utilisent par ailleurs.

Pour autant, ces systèmes ne répondent pas directement à la même problématique que la contribution que je présente dans la section suivante. D'abord, ils intègrent des bases de données relationnelles, pas des ontologies [8]. Ensuite, si on considère la classification de [WVV+01] portant sur les systèmes d'intégration à base d'ontologies ("mono-ontologie", "multi-ontologies" et "hybrides", illustrés en figure 3.6), ce sont des systèmes "mono-ontologie", et en tant que tels ils souffrent des défauts de ce type de systèmes, en particulier la difficulté notoire de construire une ontologie globale à plusieurs sources qui *fasse consensus*. Enfin, et ce n'est pas la moindre limitation, les systèmes OBDA pré-cités

5. http ://www.dis.uniroma1.it/ quonto/
6. "Minimales" dans le sens "contenant le minimum possible d'unions de requêtes".
7. http ://ontop.inf.unibz.it/
8. [CGL+08], page 29 : "We finally point out that MASTRO-I addresses the problem of data integration, and not the one of schema or ontology integration."

ne prennent pas en charge la distribution des sources au niveau du médiateur sémantique (TBox globale), mais ils s'appuient sur des solutions existantes de *bases de données relationnelles fédérées* afin de ne coupler leur TBox globale qu'à "une" base les représentant toutes[9], ce qui est certainement efficace dans un système d'information d'entreprise mais ne convient pas au contexte pour lequel nous avons conçu notre contribution, contexte que je caractérise précisément dans la section suivante.

3.2 Un médiateur sémantique générique et dynamique

Dans le chapitre "Conceptual Modeling for Data Integration" [CGL+09], l'aspect *systèmes d'intégration de données* des systèmes OBDA est analysé. L'intégration de données [Len02, RR04] est un processus qui vise à combiner des données issues de sources différentes afin de permettre leur exploitation à travers une interface d'interrogation uniforme : le schéma global. Un système d'intégration est un triplet $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, où $\mathcal{G}$ est le schéma global, $\mathcal{S}$ est l'union des schémas sources et $\mathcal{M}$ est l'ensemble d'assertions de mapping de l'un à l'autre. Il y a deux grandes approches, (i) laisser les données dans leur source, c'est l'approche *médiateur* ou (ii) extraire les données et les charger dans un entrepôt respectant le schéma global, c'est l'approche *entrepôt*.

Dans un système OBDA, l'ontologie ou plus précisément sa TBox, correspond au schéma global $\mathcal{G}$ d'un système d'intégration de données de type médiateur. Les avantages qu'il y a à utiliser un *modèle conceptuel* comme schéma global plutôt qu'un *modèle logique* (en général un schéma relationnel) comme dans l'essentiel des systèmes d'intégration existants sont les suivants :

- Le modèle conceptuel est une description du domaine d'intérêt des utilisateurs, indépendante des systèmes de gestion des données des sources. Il a une représentation graphique intuitive, qui peut être support de différents systèmes de visualisation des données et de navigation.
- C'est une approche déclarative de l'intégration de données, qui permet une réutilisation des connaissances. Elle répond clairement à un besoin *d'interopérabilité sémantique* : $\mathcal{M}$ est un ensemble de *mappings sémantiques* entre les schémas logiques des sources et le schéma conceptuel global.
- Les schémas logiques de chaque source restent utilisés dans les sources pour leurs points forts : optimisations du stockage et optimisation de l'évaluation de requêtes.
- Le composant $\mathcal{M}$ (définition des assertions de mapping) facilite l'évolutivité du système d'intégration : les schémas sources peuvent évoluer, le schéma global aussi, ce sont ces définitions qui changeront.
- Le coût de l'intégration est réduit par cet aspect incrémental (chaque source est ajoutée via de nouveaux mappings) et extensible (le schéma global peut être enrichi et les mappings seront mis à jour).

Dans notre proposition, *nous intégrons des connaissances (TBox) et non pas des données* (cf. figure 3.1). Pour autant, les avantages qui viennent d'être cités, en termes d'évolutivité, d'incrémentalité et d'extensibilité du système n'en sont pas moins vérifiés.

9. [PLL+13], page 2 : "the OBDA system comprises a single relational (SQL) data source".

En général dans un système médiateur $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, on commence par concevoir les schémas $\mathcal{G}$ et $\mathcal{S}$ avant de définir leurs correspondances dans $\mathcal{M}$. Les deux approches les plus utilisées sont GAV (global-as-view, comme dans [BBC+00]) et LAV (local-as-view, comme dans [GLR00]) [Len02, RR04]. Un mapping GAV permet d'associer à chaque élément de $\mathcal{G}$ une vue sur les sources, tandis qu'un mapping LAV permet d'associer à chaque source une vue sur $\mathcal{G}$. Les systèmes OBDA présentés précédemment sont des systèmes GAV. L'efficacité du processus d'interrogation dans le système d'intégration diffère selon l'approche. En effet, les requêtes des utilisateurs étant formulées dans les termes du schéma global $\mathcal{G}$, il s'agit de trouver les sources pertinentes pour répondre à ces requêtes.

Cela s'effectue très simplement dans le cas de mapping GAV, où il suffit de remplacer les atomes de la requête, exprimés sur le schéma global, par leur correspondance dans les sources exprimées directement dans les assertions de mapping. La ré-écriture est plus complexe dans une approche LAV, l'une des solutions les plus utilisées étant l'algorithme Minicon [PH01]. En contrepartie, la mise-à-jour du système $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ lors de l'ajout et du retrait de sources ne modifie pas $\mathcal{G}$ dans une approche LAV alors que cela peut nécessiter de reconstruire $\mathcal{G}$ dans une approche GAV, lorsqu'il consiste en un schéma logique défini à partir des schémas logiques des sources.

Pour mettre en œuvre nos contributions, nous nous plaçons dans un contexte d'intégration *sémantique* ayant des caractéristiques précises, que je résume dans la définition suivante :

Définition 3.2.1 - *contexte d'application du système médiateur :*
Le contexte d'application du système médiateur proposé doit vérifier les caractéristiques suivantes :

1. Le système d'intégration souhaité est de type portail d'accès commun, aussi bien pour des applications que via des formulaires d'interrogation ou autres systèmes de visualisation et navigation destinés à des utilisateurs.
2. Chaque source conserve un fonctionnement autonome indépendant du système d'intégration.
3. Chaque source fournit un modèle conceptuel de ses données, ou du moins de la partie de ses données qu'elle souhaite rendre accessible. Ce modèle conceptuel est exprimé par une TBox ou concrètement en OWL [10] et il sert de point d'accès aux données locales, par exemple sous la forme d'un système OBDA tel Ontop [RMC12] ou MASTRO [GLL+12] installé sur la source pour réaliser un "SPARQL endpoint" [11].

Pour un tel contexte, nous définissons un système de médiation sémantique $\mathcal{J} = \langle \mathcal{G}, \mathcal{M}, \mathcal{S} \rangle$ où $\mathcal{G}$, le schéma global représenté par une TBox, contient l'ensemble $\mathcal{S}$ des schémas sources représentés également par des TBox. Plus précisément, le schéma global est une TBox qui contient celles des sources, reliées par une taxonomie extraite d'une ontologie de référence. Puisque le schéma global contient les schémas des sources, il y a un mapping LAV implicite. Un mapping GAV simple reliant les concepts de la taxonomie à des concepts des sources y est également présent. La combinaison des deux permet une

10. Précisément en OWL 2 QL.
11. Service web acceptant des requêtes en SPARQL et retournant les réponses dans le format défini par le standard SPARQL.

ré-écriture des requêtes efficace et une mise à jour du système très souple en cas d'ajout ou de retrait d'une source. De ce fait, le système médiateur que nous proposons présente les trois caractéristiques suivantes :

1. Il s'applique dans le contexte spécifié par la définition 3.2.1.
2. L'investissement nécessaire en ressources humaines pour la construction du système d'intégration est limité [12].
3. Chaque source peut rejoindre ou quitter le système médiateur sans nécessiter de modifications complexes ni côté source [13], ni côté médiateur [14].

Plus généralement, notre contribution consiste en une construction automatique supervisée (nécessitant une intervention humaine très réduite) d'un tel système médiateur fonctionnant dans un contexte de type web sémantique, qui peut aussi bien être au sein d'un intranet que sur Internet. Grâce à sa construction incrémentale et à son adaptation aux principes OBDA, ce système médiateur a les avantages d'être dynamique et aisément interogeable. De plus, notre proposition est générique, dans le sens où elle est paramétrée par une ontologie de référence, représentant le domaine dans lequel le système de médiation est souhaité.

Je présente dans la section 3.2.1 le médiateur, dans la section 3.2.2 comment il met en œuvre les principes OBDA pour le traitement des requêtes posées sur le schéma global et dans la section 3.2.3 le caractère générique de la proposition, en donnant les grandes lignes de la construction et la maintenance incrémentales d'un tel médiateur. Ces propositions sont l'objet de la thèse de Cheikh Niang [Nia13], soutenue le 5 juillet 2013, déjà publiées pour partie [NBL10, NBLS11b, NBLS11a, NBLS12, NBLS13, NBSL13].

3.2.1 Caractéristiques du médiateur

La figure 3.3 montre le schéma global sur lequel repose le médiateur, sur un exemple formel avec 2 sources que nous appellerons $S1$ et $S2$. Dans sa partie haute se trouve la taxonomie qui relie les différentes sources, la relation de subsomption y étant représentée par un simple trait, le concept le plus général étant au-dessus. J'y ai représenté également le fait que les classes B et D sont disjointes car ce type de contrainte peut exister au niveau de la taxonomie comme au niveau des TBox sources. Ensuite, on voit dans cette figure que chaque source est représentée dans le schéma global par 2 éléments : d'une part (en bas) la TBox de la source et d'autre part un ensemble d'assertions de mappings de la forme $G_i \sqsubseteq G$, où G_i est un concept de la source S_i et G un concept de la taxonomie. La table 3.2 reprend le contenu du schéma global illustré en figure 3.3, en notation $DL\text{-}Lite_{\mathcal{A}}$.

Comme l'illustrent la figure 3.3 et la table 3.2, le médiateur dispose d'un schéma global qui est une TBox de la logique $DL\text{-}Lite_{\mathcal{A}}$, au sein de laquelle on peut distinguer plusieurs TBox, ce que précise la définition suivante.

Définition 3.2.2 - *composition du schéma global $\mathcal{T}_g$ du médiateur :*
$\mathcal{T}_g$ est une TBox de la logique $DL\text{-}Lite_{\mathcal{A}}$ composée des TBox suivantes :

12. Ceci est démontré en section 3.2.3.
13. En principe aucune modification n'est nécessaire côté source, si ce n'est éventuellement de fermer son point d'accès SPARQL lorsqu'il ne sert qu'à ce système médiateur-là.
14. Ceci est démontré en section 3.2.3.

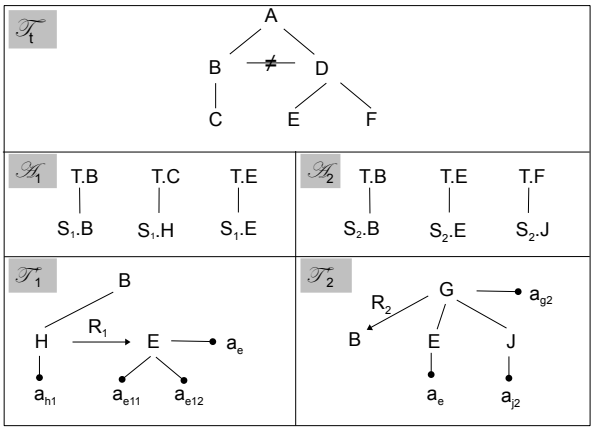


FIGURE 3.3 – Illustration de l'architecture du schéma global par un exemple

- **Une TBox** $\mathcal{T}_t$ représentant une taxonomie, TBox *atomique* dans le sens où elle ne contient que des inclusions atomiques positives (IAP) (de la forme $A_1 \sqsubseteq A_2$) et des inclusions atomiques négatives (IAN) (de la forme $A_1 \sqsubseteq \neg A_2$, où A_1 et A_2 sont des concepts atomiques). $\mathcal{T}_t$ respecte la propriété suivante :
  si B est un concept de $\mathcal{T}_t$ alors B subsume (directement ou par transitivité) au moins un concept d'une source S_i.
- Pour chaque source S_i, **une TBox** $\mathcal{T'}_i$ représentant son schéma conceptuel, TBox de la logique $DL\text{-}Lite_{\mathcal{A}}$ sans restriction particulière.
- Pour chaque source S_i, **une TBox** $\mathcal{A}_i$ composée d'un ensemble d'assertions de la forme $B_i \sqsubseteq B$ où B_i est un concept de $\mathcal{T'}_i$ et B est un concept de $\mathcal{T}_t$ (alignement).

Ce schéma global a plusieurs caractéristiques importantes à noter. Tout d'abord, la taxonomie $\mathcal{T}_t$ peut être composée de plusieurs arborescences si c'est le cas pour la taxonomie de l'ontologie de référence dont elle est extraite. En effet, les concepts de $\mathcal{T}_t$ ne sont des concepts d'aucune des sources, ils proviennent d'une ontologie de référence comme cela sera détaillé dans la section 3.2.3. Nous verrons que, par construction, tout concept de cette taxonomie est là car il est parent ou ancêtre (via les mappings) d'un concept appartenant à une source. De ce fait, si une requête porte sur un concept appartenant à $\mathcal{T}_t$ alors il y a au moins une source qui peut fournir une réponse. Par ailleurs, la taxonomie est une simple hiérarchie de concepts, lesquels n'ont ni attributs, ni rôles, donc tout atome de requête sur la partie taxonomie est forcément unaire.

Ensuite, concernant les assertions contenues dans l'ensemble des $\mathcal{A}_i$, on voit qu'elles n'impliquent elles aussi que des concepts (pas de rôles ni d'attributs dans ces parties du schéma global) et qu'elles relient une partie des concepts de la taxonomie $\mathcal{T}_t$ avec une partie des concepts de chaque TBox source $\mathcal{T'}_i$. Ce sont des alignements qui jouent le rôle des "mappings" des systèmes d'intégration, c'est-à-dire les liens entre schéma global et sources. Il est important de noter qu'ici ces mappings associent à chaque élément de la taxonomie $\mathcal{T}_t$ une union d'éléments de sources atomiques, en d'autres termes jamais une conjonction

$B \sqsubseteq A$ $C \sqsubseteq B$	$D \sqsubseteq A$ $E \sqsubseteq D$	$B \sqsubseteq \neg D$ $F \sqsubseteq D$
$B_1 \sqsubseteq B$ $H_1 \sqsubseteq C$ $E_1 \sqsubseteq E$		$B_2 \sqsubseteq B$ $E_2 \sqsubseteq E$ $J_2 \sqsubseteq F$
$H \sqsubseteq B$ $\delta(a_{h1}) \sqsubseteq H$ $\rho(a_{h1}) \sqsubseteq xsd:string$ $\exists R_1 \sqsubseteq H$ $\exists R_1^- \sqsubseteq E$ $\delta(a_{e11}) \sqsubseteq E$ $\delta(a_{e12}) \sqsubseteq E$ $\delta(a_e) \sqsubseteq E$ $\rho(a_{e11}) \sqsubseteq xsd:string$ $\rho(a_{e12}) \sqsubseteq xsd:string$ $\rho(a_e) \sqsubseteq xsd:string$		$\exists R_2 \sqsubseteq G$ $\exists R_2^- \sqsubseteq B$ $\delta(a_{g2}) \sqsubseteq G$ $\rho(a_{g2}) \sqsubseteq xsd:string$ $E \sqsubseteq G$ $J \sqsubseteq G$ $\delta(a_e) \sqsubseteq E$ $\delta(a_{j2}) \sqsubseteq J$ $\rho(a_e) \sqsubseteq xsd:string$ $\rho(a_{j2}) \sqsubseteq xsd:string$

Table 3.2 – TBox globale $\mathcal{T}_g$ correspondant à la figure 3.3.

n'apparaît dans ces mappings. De fait, l'union $\mathcal{T}_t \cup \mathcal{A}$ [15] forme une taxonomie dont *toutes les feuilles sont des concepts appartenant à une source.* Cela découle de la propriété vérifiée par $\mathcal{T}_t$ énoncée dans la définition 3.2.2. La figure 3.4 montre $\mathcal{T}_t \cup \mathcal{A}$ pour l'exemple donné en figure 3.3.

Enfin, dans chaque TBox $\mathcal{T}'_i$, il y a des concepts, des rôles et des attributs, sans restriction particulière si ce n'est d'être tous associés à la source S_i. Il est possible d'interroger chacun de ces éléments dans une requête sur le schéma global.

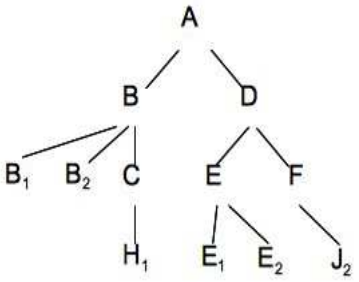


Figure 3.4 – $\mathcal{T}_t \cup \mathcal{A}$ pour l'exemple donné en figure 3.3

Nous pouvons maintenant préciser notre système médiateur $\mathcal{J} = \langle \mathcal{G}, \mathcal{M}, \mathcal{S} \rangle$ dans la définition suivante.

Définition 3.2.3 - *définition du médiateur :*
Le système médiateur $\mathcal{J} = \langle \mathcal{G}, \mathcal{M}, \mathcal{S} \rangle$ est tel que :

15. Où $\mathcal{A}$ est l'union de tous les $\mathcal{A}_i$.

- $\mathcal{G} = \mathcal{T}_g$, avec $\mathcal{T}_g$ tel qu'en définition 3.2.2.
- $\mathcal{M} = \mathcal{M}_{\mathcal{GAV}} \cup \mathcal{M}_{\mathcal{LAV}}$, où :
  - $\mathcal{M}_{GAV}$ est représenté par la taxonomie $\mathcal{T}_t \cup \mathcal{A}$ (avec $\mathcal{T}_t$ et $\mathcal{A} = \bigcup \mathcal{A}_i$ tels qu'en définition 3.2.2) qui, à tout atome de requête portant sur l'alphabet de $\mathcal{T}_t$, fait correspondre via la relation de subsomption un ou plusieurs d'atomes portant sur des concepts dans un ou plusieurs $\mathcal{T'}_i$, chacun représentant une source. Les assertions de mappings (implicites) sont de la forme $\Phi(x) \rightsquigarrow_s \Psi(x)$ où $\Psi(x)$ est une union de requêtes unaires $\varphi(x)$ portant sur un concept d'un ou plusieurs $\mathcal{T'}_i$ et l'indice s dénote la sémantique du mapping [16].
  - $\mathcal{M}_{LAV}$ est représenté par la présence des $\mathcal{T'}_i$ dans $\mathcal{T}_g$ qui, à tout atome de requête portant sur l'alphabet d'une source S_i, fait correspondre ce même atome dans $\mathcal{T'}_i$ et éventuellement aussi dans d'autres $\mathcal{T'}_k$. Les assertions de mappings (implicites) sont de la forme $\Phi(\vec{x}) \rightsquigarrow_c \Psi(\vec{x})$ où $\Psi(\vec{x})$ est une union de requêtes $\varphi(\vec{x})$ portant sur un concept, rôle ou attribut d'un ou plusieurs $\mathcal{T'}_i$.
- $\mathcal{S} = \bigcup \mathcal{T'}_i$, avec $\mathcal{T'}_i$ tels qu'en définition 3.2.2.

Il découle de la définition 3.2.3 que $\mathcal{T}_g$ contient de fait chaque partie du système médiateur. Dans notre exemple, pour illustrer $\mathcal{M}_{GAV}$ on peut constater en figure 3.4 qu'à un atome de requête portant sur le concept A vont correspondre les atomes portant sur B, H, E de $\mathcal{T'}_1$ et sur B, E, J de $\mathcal{T'}_2$. Par ailleurs concernant $\mathcal{M}_{LAV}$, à un atome de requête portant sur l'attribut a_e vont correspondre des atomes de S_1 et de S_2.

3.2.2 Interrogation des sources via le médiateur

Considérons précisément les requêtes qui peuvent être formulées sur un tel schéma global. Conformément à la première caractéristique du contexte dans lequel nous nous plaçons (spécifié dans la définition 3.2.1), un utilisateur du système médiateur peut voir le schéma global sous une forme ou une autre, dans son entier, mais en pouvant aussi savoir quelle source il interroge. Les questions d'IHM [17] que cela soulève ne sont pas abordées ici, je donne juste en figure 3.5 une illustration imaginée pour notre exemple déjà présenté figure 3.3 et table 3.2. La taxonomie $\mathcal{T}_t$ y est délimitée en gris, la source $\mathcal{T'}_1$ en bleu, la source $\mathcal{T'}_2$ en jaune, les mappings de $\mathcal{A}_1$ en bleu et les mappings de $\mathcal{A}_2$ en orange.

D'une manière générale, on remarque qu'il peut y avoir des concepts homonymes dans les différentes sources et la taxonomie, qui sont distingués par leur provenance, comme les concepts B et E dans notre exemple. C'est évidemment le cas aussi pour les rôles et les attributs (cf. a_e dans notre exemple). Le fait d'avoir le même nom dans deux sources différentes ne veut en général pas dire qu'il s'agit d'exactement la même chose, même si le système médiateur se concentre sur un domaine particulier, ce qui réduit significativement les risques d'ambiguité (sans les éliminer). Si de plus deux concepts homonymes sont subsumés par un même concept de la taxonomie alors nous faisons l'hypothèse qu'un

16. Cf. Les différentes sémantiques possibles pour les mappings : *sound* (consistante : les réponses à la requête représentant la tête du mapping sont incluses dans les réponses à la requête correspondant au corps), *complete* (complète : les réponses à la requête représentant le corps du mapping sont incluses dans les réponses à la requête correspondant à la tête) et *exact* (consistante et complète).

17. Interface Homme Machine, en tant que discipline.

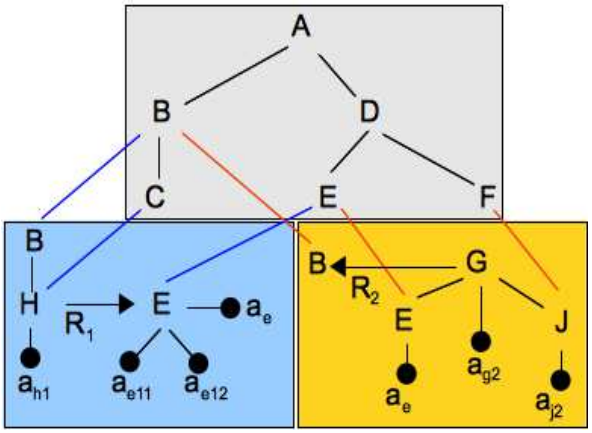


FIGURE 3.5 – Schéma de la figure 3.3 pour un utilisateur du système de médiation.

utilisateur interrogeant l'un sera satisfait de recevoir également des réponses calculées via l'autre (avec l'information de provenance).

On peut aussi penser que ces concepts de sources différentes reliés par une relation de parent commun ou d'ancêtre commun dans la taxonomie sont trop faiblement couplés et qu'il serait dans certains cas plus pertinent d'établir entre eux une relation d'équivalence. Cela fait clairement partie des perspectives de ce travail que de rechercher toutes les extensions possibles au plus petit dénominateur commun que représente la taxonomie dans le schéma global de notre système médiateur et les mappings de type inclusion des concepts des sources vers ceux de la taxonomie.

Considérons donc les différents cas d'utilisation du schéma global que nous venons de présenter, par un utilisateur qui veut interroger les sources. Nous les avons regroupés en 5 catégories listées dans la définition suivante.

Définition 3.2.4 - ***scenarios d'interrogation du schéma global $\mathcal{T}_g$ du médiateur :***

(A) L'utilisateur interroge uniquement via les concepts de la taxonomie.
(B) L'utilisateur interroge des concepts, rôles et attributs qui apparaissent tous dans une même source, en précisant qu'il veut les réponses de cette source uniquement.
(C) L'utilisateur interroge des concepts, rôles et attributs qui apparaissent tous dans une même source, sans préciser qu'il souhaite limiter les réponses à cette source.
(D) L'utilisateur interroge des concepts, rôles et attributs qui n'apparaissent pas tous dans la même source.
(E) L'utilisateur interroge des concepts de la taxonomie, avec des rôles et attributs (qui, eux, sont dans des sources).

En considérant notre exemple en figure 3.5, dans le scenario (A), si l'utilisateur interroge sur le concept A par la requête $q(x) \leftarrow A(x)$ [18] alors il doit obtenir les réponses apportées

18. Cf. section 3.1.3. Nous omettons dans la suite les symboles existentiels pour les variables liées, c'est-

par B, H, E de $\mathcal{T}'_1$ et par B, E, J de $\mathcal{T}'_2$. De même, s'il interroge sur E alors il doit obtenir les réponses apportées par E de $\mathcal{T}'_1$ et par E de $\mathcal{T}'_2$. Dans le scenario (B), la requête est transmise à la source sans ré-écriture.

Dans le scenario (C), l'utilisateur doit bien sûr recevoir les réponses de la source contenant tous les concepts, rôles et attributs interrogés, mais il est possible qu'il en obtienne également provenant d'autres sources. Par exemple, s'il interroge H, R_1 et E, étant donné que E est dans $\mathcal{T}'_1$ mais aussi dans $\mathcal{T}'_2$ et $\mathcal{T}_t$, il pourra recevoir des réponses, partielles, en provenance de $\mathcal{T}'_2$. Dans le scenario (D), par exemple si l'utilisateur interroge H, G et a_{h1} alors il pourra recevoir des réponses de S_1 (interrogée via $\mathcal{T}'_1$) concernant H et a_{h1}, et de S_2 (interrogée via $\mathcal{T}'_2$) concernant G. Remarquons que le fait que certaines instances de G et certaines instances de H puissent être en relation via les trois concepts B ne peut être inféré automatiquement à partir des seules relations de subsomption : c'est un cas entrant dans la perspective évoquée précédemment, d'enrichissement des contraintes contenues dans le schéma global obtenu par notre proposition générique décrite en section 3.2.3.

Enfin dans le scenario (E), par exemple si l'utilisateur interroge D et les attributs a_e et a_{j2}, alors il pourra recevoir des réponses de S_1 (interrogée via $\mathcal{T}'_1$) concernant ses instances de E avec l'attribut a_e et de S_2 (interrogée via $\mathcal{T}'_2$) concernant ses instances de E avec l'attribut a_e et ses instances de J avec l'attribut a_{j2}.

Notons le fait qu'il est peu probable qu'il y ait d'entrée de jeu dans S_1 et dans S_2 une même instance du concept E : en cas de système OBDA les instances sont construites à partir des valeurs (en partiiculier les identifiants) contenues dans les bases sources par un mécanisme résolvant le problème dit d'"impedance mismatch" [CLL+06, CGL+07], cf. section 3.1.3. C'est l'utilisateur, par des requêtes successives, qui peut déterminer des rapprochements plus précis que ce qu'expriment les relations de subsomption des alignements $\mathcal{A}_i$. Le problème de la *résolution d'entités* [GM12] est ainsi une perspective riche pour ce travail : comment, dans un tel système médiateur, déterminer automatiquement des entités similaires.

Etant donnée une requête conjonctive $q(\bar{x}) \leftarrow conj(\bar{x}, \bar{y})$ posée sur le schéma global $\mathcal{T}_g$, la résolution de q suit les étapes listées dans la définition 3.2.5. Remarquons que par la suite nous distinguons dans une requête conjonctive les atomes qui portent sur la taxonomie de ceux qui portent sur une source. Nous appelons les premiers *"atomes sur taxonomie"* (ils sont toujours unaires) et les derniers *"atomes sur source(s)"* (unaires ou binaires).

Définition 3.2.5 - *étapes d'évaluation de la requête globale :*
Etant donnée une requête conjonctive $q(\bar{x}) \leftarrow conj(\bar{x}, \bar{y})$ posée sur le schéma global $\mathcal{T}_g$, les étapes de traitement de q sont, dans l'ordre :

1. Application de l'algorithme Consistent [CGL+07] avec q vue comme une instance canonique, pour vérification de la consistance de q par rapport à $\mathcal{T}_g$. Si la requête est inconsistante par rapport aux contraintes dans $\mathcal{T}_g$ alors retour d'un résultat vide.
2. Application de l'algorithme PerfectRef [CGL+07] sur q et $\mathcal{T}_t \bigcup \mathcal{A}$, puis sélection des parties du résultat à conserver pour l'étape suivante. Le résultat de PerfectRef est une union de requêtes conjonctives et nous ne conservons dans cette union que les requêtes conjonctives entièrement composées *d'atomes sur sources.*

à-dire celles qui ne sont pas dans $q(\bar{x})$, la *tête* de la requête.

3. Etant donnée Q_s l'union de requêtes obtenue à l'étape 2, calcul des requêtes à envoyer à chaque source par une adaptation de l'algorithme MiniCon [PH01].
4. Envoi des requêtes à chaque source pour obtenir les réponses des sources et production de la réponse globale.

La première étape est destinée à ne pas évaluer des requêtes qui ne peuvent avoir de résultat étant données les contraintes de disjonction du schéma global. Dans notre exemple, sachant que $B \sqsubseteq \neg D$, il est inutile d'évaluer la requête $q(x) \leftarrow B(x), D(x)$. Nous avons rappelé en section 3.1.3 que l'algorithme Consistent a une complexité polynomiale sur la taille de la TBox, ici $\mathcal{T}_g$, et sur la taille de la ABox, ici le nombre d'atomes de q.

La deuxième étape permet d'évaluer comme décrit précédemment les requêtes du scenario (A) et, en partie, celles des scenario (C), (D) et (E). Nous avons expliqué en section 3.1.3 que l'algorithme PerfectRef permet de compiler la connaissance contenue dans la TBox dans la requête elle-même : ici cette connaissance est uniquement dans les relations de subsumption qui forment la taxonomie $\mathcal{T}_t \cup \mathcal{A}$ contenue dans $\mathcal{T}_g$. Grâce à cette connaissance, les atomes exprimés sur la taxonomie sont ré-écrits en atomes exprimés sur *les feuilles* de $\mathcal{T}_t \cup \mathcal{A}$, or ces feuilles sont nécessairement des atomes de sources. Cela découle de la spécification de $\mathcal{T}_t$ énoncée dans la définition 3.2.2. Nous avons rappelé en section 3.1.3 que l'algorithme PerfectRef a une complexité polynomiale sur la taille de $\mathcal{T}_t \cup \mathcal{A}$ et exponentielle sur la taille de q. Nous avons également indiqué que les versions récentes de PerfectRef sont optimisées pour réduire considérablement en pratique la complexité sur la taille de la requête, en évitant quantité de reformulations qui sont en fait contenues dans d'autres.

Notons (1) qu'appliquer PerfectRef sur $\mathcal{T}_g$ tout entier ne permet pas de "remonter" d'une $\mathcal{T'}_i$ à $\mathcal{T}_t$ pour redescendre sur une autre, car les relations de mappings sont des relations de subsomption et non d'équivalence, et (2) que PerfectRef est probablement appliqué au sein de chaque source, lesquelles sont en général des systèmes OBDA d'après la caractéristique 3 du contexte d'application, cf. définition 3.2.1. Ces deux faits impliquent que nous obtenons le même résultat en utilisant à cette étape seulement $\mathcal{T}_t \bigcup \mathcal{A}$ plutôt que $\mathcal{T}_g$ tout entier, réduisant de ce fait la complexité de l'étape.

La troisième étape du processus permet, avec l'étape 2, de répondre aux cas d'utilisation C), D) et E). A ce stade nous avons des reformulations qui n'impliquent que les parties du schéma global qui sont des sources : sur ces parties-là c'est un mapping LAV que nous avons (cf. la décomposition de $\mathcal{M}$ dans la définition 3.2.3). Si on interroge par exemple sur un nom de personne, il peut y avoir dans le schéma global plusieurs concepts *Personne*, ayant un attribut *nom* : le schéma global contient cette connaissance, il reste à déterminer (automatiquement bien entendu) toutes les sources ayant ce concept et cet attribut et à construire les requêtes qui doivent leur être envoyées. Nous appliquons donc un calcul de type MiniCon [19] [PH01] pour chacune des requêtes conjonctives contenues dans l'union de requêtes conjonctives Q_s issues de l'étape 2. Ce calcul prend en entrée une requête *composée exclusivement d'atomes sur sources* et en général d'atomes *sur plusieurs sources* (dans notre exemple c'est le cas pour B, E, a_e, etc.). Il détermine les sources impliquées ainsi que les requêtes qui doivent leur être envoyées. Comme dans MiniCon, il forme d'abord des

19. L'un des algorithmes pour mappings LAV les plus efficaces en pratique, y compris à grande échelle, comme rappelé dans [AMR+12].

buckets [LRO96] regroupant les sources pouvant donner des réponses pour chaque atome g de q. Les différents buckets sont ensuite combinés afin de construire les sous-requêtes valides pour les différentes sources. Nous veillons à ce que la validité des sous-requêtes tienne compte ici des contraintes exprimées dans les $\mathcal{T}'_i$ (contraintes qui sont inexistantes dans le cadre d'application du MiniCon classique). L'algorithme est présenté en détails dans la thèse de Cheikh Niang [Nia13].

Concernant la complexité, la création des buckets pour l'ensemble des requêtes $q_{\mathcal{S}}$ contenues dans Q_s est au pire en $O(n \times l \times k \times t)$, où n est le nombre de requêtes dans Q_s, l est le nombre moyen d'atomes dans les requêtes $q_{\mathcal{S}}$, k est le nombre de sources et t est la taille moyenne des $\mathcal{T}'_i$. La combinaison des buckets est en $O(n' \times b^l)$, où n' est nombre de requêtes $q_{\mathcal{S}}$ pour lesquelles des buckets ont pû être créés pour tous leurs atomes (donc $n' \leq n$), l est le nombre moyen d'atomes de ces requêtes et b est la taille moyenne des buckets créés.

La dernière étape est réalisée en utilisant les points d'accès SPARQL des sources (cf. caractéristique 3 de la définition 3.2.1). Le résultat global est calculé par le médiateur à partir des résultats retournés par les sources (ces résultats sont des ensembles de n-uplets, sur lesquels sont opérées les jointures nécessaires) avant de pouvoir être affiché à l'utilisateur. Les étapes 1 à 3 sont détaillées dans la thèse de Cheikh Niang, elles ont été implémentées dans le cadre de son application [Nia13] et publiées en partie [NBLS12, NBLS13]. Elles font également l'objet d'une mise en oeuvre dans le cadre du projet Personae dont il sera question en section 3.3, et pour lequel l'étape 4 sera réalisée pour la première fois.

J'ai montré dans cette section comment le système médiateur qui vient d'être décrit réalise efficacement son rôle de portail commun d'accès aux sources, dans le contexte spécifié dans la définition 3.2.1. Je mets en évidence dans la section suivante les deux autres atouts importants de ce médiateur, à savoir (1) qu'il peut être construit quasi-automatiquement, et (2) que l'ajout et le retrait d'une source ont un impact très limité sur son schéma global $\mathcal{G}$. Ce sont deux atouts importants dans le contexte du web, où une source peut rejoindre le système intégré le temps d'échanger ses données ou de contribuer à une tâche particulière, et peut aussi se retirer.

3.2.3 Construction et maintenance incrémentales

Notre système nécessite que les sources fournissent un schéma conceptuel et permettent d'interroger leurs données via ce schéma conceptuel comme le font les systèmes OBDA. A partir de là nous avons mis au point une méthode pour construire quasi-automatiquement un médiateur comme celui présenté dans la section précédente. L'idée principale est de s'appuyer pour cela sur une ontologie de référence, qui va apporter le minimum de connaissance nécessaire pour structurer les liens entre les schémas des sources, cette connaissance étant représentée par la TBox $\mathcal{T}_t$, taxonomie du schéma global telle qu'introduite dans la définition 3.2.2.

En effet, la difficulté principale dans le processus de construction de systèmes médiateurs réside dans la définition du schéma global, qui doit représenter (au moins) les sources. Cette difficulté récurrente est déjà décrite en 2001 dans le fameux état de l'art sur les dif-

férentes approches d'intégration de données via des ontologies [WVV+01], qui classifie ces approches comme rappelé en figure 3.6. En synthèse (cf. figure 3.6), les approches mono-ontologies butent sur cette difficulté qu'il y a à établir (et à maintenir au fil des évolutions) un consensus sur l'ontologie globale. Les approches multi-ontologies, qui correspondent aux systèmes PDMS [20] actuels [HIST05], évitent cet écueil mais les mappings qu'il faut établir et maintenir entre les sources nécessitent une quantité non négligeable d'expertise humaine. L'approche hybride a été proposée pour faciliter l'intégration de sources et l'interrogation des systèmes multi-ontologies. Par rapport à cette classification notre proposition se rapproche de ce dernier cas de figure en allant plus loin : il fédère les différents mappings qui peuvent exister entre les ontologies de chaque source dans l'ontologie globale (qui n'est plus un simple vocabulaire), cela à l'aide d'une ontologie de référence qui est supposée faire d'ores et déjà l'objet du consensus nécessaire [21].

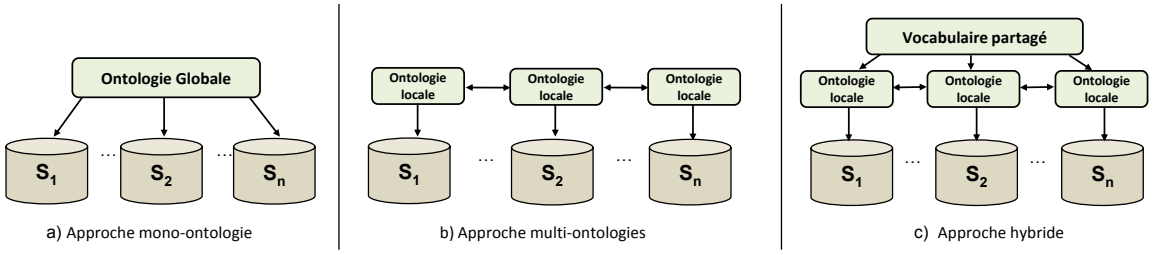


Figure 3.6 – Classification des systèmes d'intégration à base d'ontologies de [WVV+01].

Disposer a priori d'une telle ontologie globale est devenu récemment envisageable du fait que dans le cadre du web sémantique les collaborations se multiplient dans de nombreux domaines et suscitent la constitution d'ontologies de référence accessibles publiquement. En réalité les efforts collaboratifs de structuration des informations et de la connaissance ne datent pas d'hier, le deuxième chapitre du présent document en a d'ailleurs présenté un cas précis concernant la représentation des ressources lexicales. Mais le web sémantique constitue une caisse de résonance qui ouvre les initiatives en cours, souvent engagées dans des démarches de standardisation, vers le grand public. Des exemples particulièrement significatifs existent par exemple dans le domaine de l'agriculture avec AGROVOC [22], ou encore dans le domaine biomédical avec SNOMED CT [23] et bien d'autres [24], etc. On trouve dans le domaine de *l'alignement d'ontologies* plusieurs propositions qui consultent une ontologie de référence pour améliorer les résultats de l'alignement de deux ontologies [AtKvH06, AKtKvH06, SdM06, RS07]. Ces travaux ont renforcé notre intuition qu'une ontologie de référence peut être le support de la construction d'un schéma global médiateur.

Nous utilisons une ontologie de référence pour construire semi-automatiquement le

20. Peer Data Management Systems, systèmes de collaborations distribuées et dynamiques entre pairs.
21. C'est une hypothèse forte qui se vérifie dans le cas d'application de la thèse de Cheikh Niang avec AGROVOC mais peut ne pas se vérifier dans bien d'autres applications.
22. http ://aims.fao.org/standards/agrovoc/about
23. http ://www.ihtsdo.org/
24. http ://bioportal.bioontology.org/ontologies/

schéma global $\mathcal{G}$ selon la démarche synthétisée dans la figure 3.7, où l'ontologie de référence utilisée est notée OM. On y voit les 2 étapes de la construction de $\mathcal{G}$ (dénoté OG) :

- Nous construisons d'abord pour chaque source un schéma intermédiaire appelé *accord* et noté A dans la figure 3.7, qui contient la partie de son schéma OL qui est effectivement concernée par le domaine d'intégration (domaine représenté par l'ontologie de référence OM). Cette phase s'appuie sur des techniques d'alignement d'ontologies, l'alignement étant entre l'ontologie source OL et l'ontologie de référence OM. Elle nécessite une supervision par un expert, pour valider et éventuellement compléter les correspondances trouvées automatiquement.
- Chaque accord est ensuite automatiquement concilié, d'une manière incrémentale, dans le schéma global OG. La conciliation est faite en liant sémantiquement les concepts locaux via une taxonomie calculée à partir de l'ontologie de référence OM. Les sources restent ainsi indépendantes les unes vis-à-vis des autres tout en étant liées sémantiquement dans le schéma global.

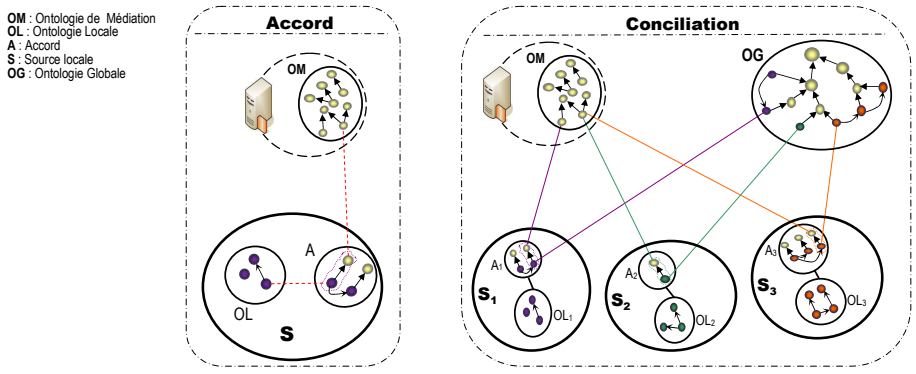


FIGURE 3.7 – Construction du schéma global [Nia13]

La construction de l'ontologie globale va mettre en jeu les TBox précisées dans la définition suivante.

Définition 3.2.6 - ***les TBox pour la construction de l'ontologie globale sont :***

- **Les TBox locales** $\mathcal{T}_i$. Chaque source S_i de la figure 3.7 est représentée par une TBox $\mathcal{T}_i$, correspondant à OL_i dans la figure.
- **La TBox de référence** $\mathcal{T}_r$. C'est la partie taxonomie de l'ontologie de référence (c'est-à-dire une partie de l'ontologie OM de la figure 3.7), c'est donc une TBox qui ne contient que des inclusions (positives ou négatives) impliquant des concepts *atomiques*.
- **Les TBox des accords** $\mathcal{T}_{ai}$. Chaque A_i de la figure 3.7 est une TBox $\mathcal{T}_{ai} = \langle \mathcal{T'}_i, \mathcal{A}_i \rangle$ composée de $\mathcal{T'}_i$, une partie de $\mathcal{T}_i$ contenant les assertions qui sont pertinentes dans le cadre du processus d'intégration, et de $\mathcal{A}_i$, l'alignement entre $\mathcal{T'}_i$ et $\mathcal{T}_r$, c'est-à-dire un ensemble d'assertions de type $B \sqsubseteq A$ où B est un concept de $\mathcal{T'}_i$ et A est un concept de $\mathcal{T}_r$).

Le résultat du processus de construction est la TBox globale (dénotée OG dans la figure 3.7) $\mathcal{T}_g = \langle \mathcal{T}_a, \mathcal{T}_t \rangle$. Elle se compose de l'ensemble des accords ($\mathcal{T}_a = \bigcup \mathcal{T}_{ai}$) et de $\mathcal{T}_t$, la plus petite portion de $\mathcal{T}_r$ qui concilie tous les $\mathcal{T}_{ai}$ dans $\mathcal{T}_g$. En d'autres termes, la TBox globale est automatiquement construite pour correspondre au schéma de médiation spécifié par la définition 3.2.2.

Détermination d'un accord

Il peut arriver qu'une source couvre un domaine plus large que celui visé par le système d'intégration. Par exemple dans le projet Personae présenté en section 3.3, c'est le cas de la base Budé [25] qui contient des informations concernant la transmission des œuvres antiques et médiévales par les manuscrits et les imprimés anciens pour une période qui couvre la fin du Moyen Âge et la Renaissance. Le projet Personae concerne la prosopographie [26] au cours de cette période, son champ d'intérêt (essentiellement les personnes et leurs carrières, déplacements, relations...) couvre donc la base Budé, qui compte actuellement entre 12 000 et 13 000 personnes, mais seulement une partie de cette base, qui comprend par ailleurs la description de milliers de manuscrits et imprimés.

L'accord doit contenir les assertions de la TBox locale qu'il sera utile d'interroger via la TBox globale. Pour déterminer ce contenu, nous appliquons d'abord un processus d'alignement qui consiste à trouver des correspondances entre les concepts atomiques de la TBox locale, appelés *concepts ancrés*, et les concepts de la TBox de référence, appelés *concepts ancres*. Ce premier processus calcule donc $\mathcal{A}_i$ telle que décrite dans la définition 3.2.6, en d'autres termes les correspondances trouvées sont dénotées par des assertions $B \sqsubseteq A$ où B est un concept local et A est un concept de référence. Une fois calculé l'ensemble des concepts ancrés, nous pouvons compléter l'accord, qui va contenir les concepts ancrés, leurs relations et leurs attributs, ainsi que certains autres concepts liés aux concepts ancrés. Ce deuxième processus calcule donc $\mathcal{T}'_i$ telle que décrite dans la définition 3.2.6, un sous-ensemble de la TBox locale $\mathcal{T}_i$. L'accord en lui-même est formé de $\mathcal{A}_i$ et de $\mathcal{T}'_i$.

Pour obtenir $\mathcal{A}_i$ nous réalisons (i) un *alignement syntaxique* qui s'appuie sur des techniques classiques de comparaison de chaînes de caractères, complété par (ii) une extension *sémantique* qui permet de sélectionner dans la TBox locale d'autres concepts pour être ancrés également. Cette sélection s'appuie sur la transitivité de la relation de subsomption pour ancrer tout concept atomique de $\mathcal{T}_i$ subsumé par un concept ancré B (le concept ancre est le même que celui de B). Rappelons que l'alignement permet de n'avoir dans l'accord que des concepts pertinents pour le domaine d'intégration.

Ensuite, pour obtenir $\mathcal{T}'_i$ nous avons défini un ensemble de règles qui s'appliquent sur $\mathcal{T}_i$ et $\mathcal{A}_i$, pour sélectionner dans la TBox locale $\mathcal{T}_i$ toutes les assertions qui ont un lien avec les concepts ancrés. Ces règles sélectionnent les attributs et les relations entre concepts ancrés, mais aussi certains concepts en relation avec des concepts ancrés (pour ne pas perdre de relation importante). Cette étape permet d'avoir dans l'accord un maximum d'assertions pertinentes pour le domaine d'intégration. Par ailleurs, une partie des règles de sélection permet de garantir une équivalence entre $\mathcal{T}'_i$ et $\mathcal{T}_i$ en termes de consistance.

25. http ://bude.irht.cnrs.fr/

26. Etude des biographies des membres d'une catégorie spécifique de la société, le plus souvent des élites, sociales ou politiques, en particulier leurs origines, leurs alliances et leurs environnements familiaux.

Les algorithmes de calcul de l'accord $\mathcal{T}_{ai} = \langle \mathcal{T'}_i, \mathcal{A}_i \rangle$ sont détaillés dans [Nia13] et ont été publiés dans [NBL10, NBLS11b, NBLS11a, NBSL13]. La complexité de ce calcul est polynomiale sur la taille (nombre de concepts) de la TBox locale et, pour l'alignement lexical, sur la taille de la TBox de référence. Les évaluations expérimentales ont confirmé cette complexité théorique. En pratique les TBox locales sont relativement petites, il s'agit d'ontologies légères, non pas descriptions riches d'une partie du monde mais plutôt descriptions simples du contenu d'une base de données. L'ontologie de référence peut par contre être très volumineuse, comme celle d'AGROVOC par exemple, qui contient 27888 concepts et leur description. Nos expérimentations avec cette dernière ont nécessité un temps de calcul de l'accord de 1 minute pour une TBox locale d'une cinquantaine de concepts et le double pour une TBox locale d'une centaine de concepts.

Le calcul de l'accord que je viens de résumer souffre des limites connues des techniques d'alignement s'appuyant sur des comparaisons de chaînes de caractères [SE05, SE08, SE13] : la seule similarité syntaxique des mots, ou des groupes de mots utilisés pour décrire les concepts, ne garantit ni de trouver tous les liens entre les deux ontologies, ni de ne trouver que des liens pertinents. De ce fait la construction de l'accord reste supervisée. Il faut qu'un utilisateur, pas forcément expert du domaine ni de l'ingénierie des connaissances, mais qui connaisse bien les données locales, puisse valider les correspondances trouvées automatiquement ou sélectionner la plus appropriée dans le cas où l'alignement lexical trouve plusieurs ancres pour un même concept local, voire enfin ajouter des correspondances non détectées automatiquement.

Cette intervention humaine est sans commune mesure avec ce que nécessitent les systèmes d'intégration de données classiques pour leur construction et mise en place, car elle ne consiste qu'à vérifier les résultats d'un alignement automatique d'une *ontologie légère*, avec intervention sur un nombre limité de concepts. Au contraire, une telle intervention humaine (assistée par l'ordinateur) introduit une garantie de qualité, qualité dont va dépendre celle de l'ontologie globale, construite à partir des accords des sources et de l'ontologie de référence.

Conciliation d'un accord dans l'ontologie globale

La conciliation est réalisée d'une manière incrémentale en intégrant les accords un par un dans $\mathcal{T}_g$. L'intégration d'un nouvel accord se fait en reliant ses concepts avec ceux déjà présents dans $\mathcal{T}_g$. Les liens entre accords sont établis via les concepts ancres, qui proviennent de l'ontologie de référence : nous reproduisons dans $\mathcal{T}_g$ les liens qui existent entre les différentes ancres dans $\mathcal{T}_r$. Le principe est illustré en figure 3.8, dans laquelle on a déjà intégré la source S_1 de l'exemple de la figure 3.3 dans $\mathcal{T}_g$. Il faut noter ici que la conciliation de la première source dans un schéma global encore vide a conduit à introduire des liens entre les concepts de S_1 qui n'existent pas dans la source elle-même, du fait de l'introduction du concept A, ancêtre commun aux concepts B et E dans la taxonomie de référence. La figure montre une étape de la conciliation de l'accord de la source S_2 dans $\mathcal{T}_g$, celle qui intègre dans $\mathcal{T}_g$ le concept J de S_2, dont l'ancre est le concept F.

Un concept source B est intégré dans le schéma global $\mathcal{T}_g$ via son ancre A en deux étapes : (i) nous recherchons dans $\mathcal{T}_g$, plus précisément dans $\mathcal{T}_t$ (partie taxonomie de $\mathcal{T}_g$) le concept ancre A' le plus proche de A et (ii) nous ajoutons dans $\mathcal{T}_g$ l'assertion $B \sqsubseteq A$,

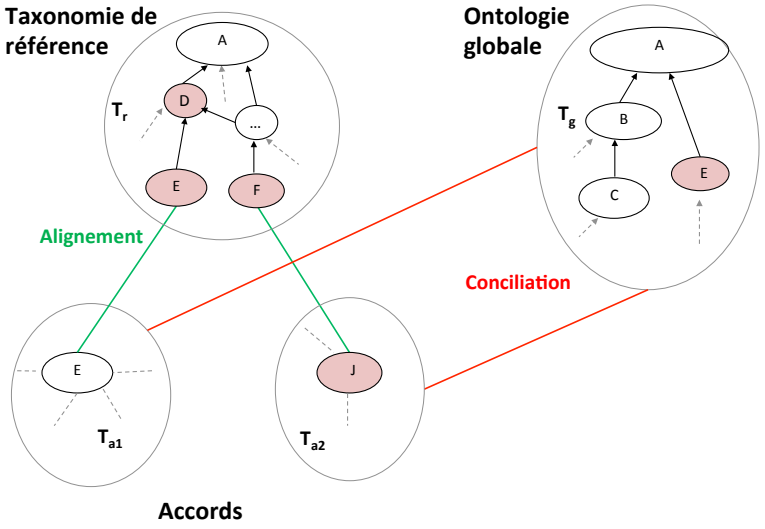


Figure 3.8 – Illustration de la conciliation

plus des liens entre A et A' que nous trouvons dans $\mathcal{T}_r$. Dans l'exemple de la figure 3.8, le concept ancre E est celui qui est le plus proche du concept ancre F et ils sont reliés dans $\mathcal{T}_r$ via D, qui est subsumé par A. De ce fait, l'ajout du concept J au schéma global consiste à y ajouter l'assertion $J \sqsubseteq F$ mais également les assertions $D \sqsubseteq A$, $E \sqsubseteq D$ et $F \sqsubseteq D$. Cela donne en résultat la taxonomie $\mathcal{T}_t$ représentée par la partie grisée en figure 3.5.

La taxonomie de référence est la base des calculs pour chacune des deux étapes. C'est sur elle qu'est définie la relation de proximité utilisée à l'étape (i) et c'est en elle qu'est recherché le plus proche ancêtre commun des deux concepts ancres considérés dans l'étape (ii). Le calcul du concept ancre (déjà dans $\mathcal{T}_t$) le plus proche de celui à intégrer se fait de la manière suivante : soit A le concept à intégrer, les concepts de $\mathcal{T}_t$ [27] les plus susceptibles d'avoir un lien de proximité avec A sont ceux qui sont dans la même hiérarchie que A dans $\mathcal{T}_r$. Nous appliquons sur cette hiérarchie une mesure de similarité structurelle afin de déterminer le concept A' de $\mathcal{T}_t$ le plus proche de A. La mesure de similarité structurelle que nous utilisons est une adaptation à notre contexte de celle proposée par Wu et Palmer [WP94]. Soit $\mathcal{H}_r$ une hiérarchie de $\mathcal{T}_r$, la proximité structurelle entre deux concepts A_1 et A_2 dans $\mathcal{H}_r$ est définie par

$$sim(A_1, A_2) = \frac{2 * depthOf(ppac_{\mathcal{H}_r}(A_1, A_2))}{depthOf(A_1) + depthOf(A_2)}$$

où $depthOf(A)$ est la fonction qui retourne la profondeur de A dans la hiérarchie $\mathcal{H}_r$ et $ppac_{\mathcal{H}_r}(A_1,A_2)$ est la fonction qui calcule le plus proche ancêtre commun aux deux concepts A_1 et A_2 dans la hiérarchie $\mathcal{H}_r$, fonction que je précise dans la suite. Nous calculons (avec cette mesure) la similarité entre le concept ancre A à ajouter et chacun des concepts déjà

27. Rappel : $\mathcal{T}_g = \langle \mathcal{T}_a, \mathcal{T}_t \rangle$ où $\mathcal{T}_t$ est la partie taxonomie, extraite de $\mathcal{T}_r$.

dans $\mathcal{T}_t$ de manière à sélectionner le concept A' de $\mathcal{T}_t$ pour lequel cette similarité est la plus grande.

Le calcul du *plus proche ancêtre commun (ppac)* de deux ancres est également réalisé sur la taxonomie de référence $\mathcal{T}_r$, qui est composée d'une ou plusieurs hiérarchies disjointes dont chacune a une structure arborescente. Dans une telle configuration, si le *ppac* de deux concepts existe, alors il est unique. Le concept C de $\mathcal{T}_r$ est le plus proche ancêtre commun aux deux concepts A_1 et A_2 dans $\mathcal{T}_r$ si et seulement si il vérifie les deux conditions suivantes :

1. $A_1 \sqsubseteq C$ et $A_2 \sqsubseteq C$ directement ou par transitivité ; et
2. pour tout concept C' de $\mathcal{T}_r$ tel que $A_1 \sqsubseteq C'$ et $A_2 \sqsubseteq C'$ on a $C \sqsubseteq C'$

Dans [BKM99a] les auteurs décrivent un calcul du *least common subsumer (lcs)* de deux descriptions de concepts en logique de description. Notre contexte est un cas particulier de celui dans lequel s'inscrit leur proposition, la structure de $\mathcal{T}_r$ étant moins complexe que celles des logiques considérées dans [CBH+92, BKM99a], qui sont très expressives (les concepts y sont définis par composition de différentes primitives avec différents opérateurs). Nous utilisons donc un calcul simplifié, qui détermine le *ppac* de deux concepts à partir de la fermeture transitive de la relation de subsomption sur $\mathcal{T}_r$. C'est un calcul quadratique sur la hauteur de $\mathcal{T}_r$.

Le coût du calcul du *ppac*, donc également de la proximité structurelle, est fonction de la taille de la TBox de référence $\mathcal{T}_r$. Ce coût peut être par conséquent élevé car en général l'ontologie de référence est de grande taille. Il est possible de partitionner la TBox de référence en blocs afin de limiter les calculs à l'intérieur des blocs contenant les concepts ancre à relier. Pour ce faire, nous avons également adapté les méthodes de partitionnement d'ontologies proposées dans la littérature [HZQ06, HSZR09]. Le but original de ces méthodes est d'améliorer l'efficacité des alignements [HSZR09]. Nous utilisons le partitionnement implanté dans le système Falcon-AO [HQ08] qui, en utilisant l'algorithme de clustering ROCK [GRS99] et en introduisant la notion de liens pondérés (selon une mesure de similarité structurelle entre concepts), permet de décomposer une ontologie en blocs *en classant dans chaque bloc les concepts les plus proches les uns des autres*. Selon les auteurs de [HZQ06], la complexité en temps de l'algorithme de partitionnement est en $O(n^2)$, n étant la taille de $\mathcal{T}_r$. Ce processus n'est exécuté qu'une seule fois et chaque bloc est sauvegardé comme une sous-partie de $\mathcal{T}_r$ dont les assertions forment une unique hiérarchie. La fermeture transitive de la relation de subsomption peut également être appliquée une seule fois sur ces blocs. L'algorithme de conciliation s'appuie alors sur les blocs issus de ces prétraitements.

L'algorithme de conciliation (intégration des accords un par un dans $\mathcal{T}_g$) et plus particulièrement l'algorithme d'intégration d'un accord dans $\mathcal{T}_g$ sont détaillés dans [Nia13] et ont été publiés dans [NBLS11b, NBLS11a, NBSL13]. L'algorithme d'intégration d'un accord dans $\mathcal{T}_g$ prend en entrée la TBox $\mathcal{T}_{ai} = \langle \mathcal{T}'_i, \mathcal{A}_i \rangle$ de l'accord d'une source, la version courante de la TBox globale $\mathcal{T}_g = \langle \mathcal{T}_a, \mathcal{T}_t \rangle$ et l'ensemble de blocs $\mathcal{T}_\mathcal{B}$ issus du partitionnement de la TBox de référence $\mathcal{T}_r$ et il retourne une nouvelle version de $\mathcal{T}_g$. Pour chaque concept ancre A appartenant à l'alignement $\mathcal{A}_i$ de $\mathcal{T}_{ai}$, il identifie d'abord le bloc $\mathcal{T}_{\mathcal{B}_k}$ de $\mathcal{T}_\mathcal{B}$ contenant A, puis il cherche parmi les concepts déjà dans $\mathcal{T}_t$ et appartenant à $\mathcal{T}_{\mathcal{B}_k}$ lequel est le plus proche de A, noté A'. Enfin il établit dans $\mathcal{T}_t$ la relation entre A et A' grâce à leur *ppac* dans $\mathcal{T}_{\mathcal{B}_k}$, calculé précédemment pour déterminer A'.

Il est clair que l'algorithme de conciliation *termine* car chaque alignement d'accord à concilier a un nombre fini de concept ancre et le calcul à réaliser pour chaque concept ancre, de type recherche dans un ensemble, termine. De plus, la notion de similarité structurelle utilisée (symétrique) et le principe d'utilisation du plus proche ancêtre commun garantissent une *indépendance du résultat par rapport à l'ordre d'ajout des accords*. La complexité en temps de l'intégration d'un nouvel accord est en $O(m \times k + (m-t) \times (b \times s + kh^2))$ où m est le nombre d'ancres de l'accord, k est la taille (nombre de concepts) de $\mathcal{T}_t$ (il faut vérifier pour chaque ancre si elle est déjà dans $\mathcal{T}_t$, auquel cas il n'y a pas de calcul supplémentaire), t est le nombre d'ancres de l'accord déjà présentes dans $\mathcal{T}_t$ (pour chaque ancre non présente dans $\mathcal{T}_t$ il faut rechercher l'ancre présente la plus proche et relier la nouvelle ancre à celle-ci), b est le nombre de blocs générés par le partitionnement de $\mathcal{T}_r$, s et h sont respectivement la taille et la hauteur moyenne des blocs. Le facteur $b \times s$ représente la recherche du bloc auquel appartient le concept ancre à intégrer et le facteur kh^2 représente la recherche du plus proche concept déjà dans $\mathcal{T}_t$ (qui intègre la recherche du *ppac*).

Il est intéressant de noter que la complexité en temps de la conciliation décroit en fonction du nombre d'accords déjà conciliés car l'augmentation du nombre d'accords conciliés augmente la probabilité de trouver des ancres déjà présentes dans la TBox globale et diminue ainsi le nombre de nouvelles ancres à intégrer (facteur $(m-t)$). Les 2 cas expérimentaux testés dans le cadre de la thèse de Cheikh Niang [Nia13], ayant l'un et l'autre 2 accords à intégrer, démontrent effectivement une diminution d'un facteur proche de 10 entre le temps d'intégration du premier accord et celui pour le deuxième accord (1 minute au lieu de 8 pour le premier cas et 2 minutes au lieu de 20 pour le second cas).

L'algorithme d'intégration d'un accord est utilisé pour la construction du médiateur à partir de plusieurs sources et c'est le même algorithme qui est utilisé pour intégrer à tout moment une nouvelle source au système médiateur.

Suppression d'une source

L'algorithme pour la suppression d'un accord $\mathcal{T}_{ai} = \langle \mathcal{T'}_i, \mathcal{A}_i \rangle$ du schéma global $\mathcal{T}_g = \langle \mathcal{T}_a, \mathcal{T}_t \rangle$ est très simple, il consiste à (i) supprimer $\mathcal{T}_{ai}$ de $\mathcal{T}_a$ et (ii) supprimer de la taxonomie$\mathcal{T}_t$ les concepts qui ne subsument plus aucun concept source. La deuxième étape est une itération ascendante sur les niveaux de $\mathcal{T}_t$, qui supprime les feuilles de $\mathcal{T}_t$ non associées à au moins un concept source et s'arrête lorsque toutes les feuilles de $\mathcal{T}_t$ sont à nouveau associées à au moins un concept source. Cette itération utilise $\mathcal{A}$, c'est-à-dire l'union des alignements, de laquelle les assertions de $\mathcal{A}_i$ ont été supprimées à l'étape (i). Nous veillons ainsi à garder dans la TBox $\mathcal{T}_t$ tous les concepts ancres de l'accord à supprimer qui sont liés à d'autres sources, ce qui implique moins de changements d'une part et une plus grande probabilité de diminuer le temps de calcul lorsque d'autres sources seront ajoutées. Ceci tout en maintenant bien la propriété de $\mathcal{T}_t$ énoncée en définition 3.2.2.

La complexité en temps de cet algorithme est au pire des cas égale à $O(a \times k)$, où a est le nombre d'assertions dans $\mathcal{A}$ et k est la taille de $\mathcal{T}_t$. On peut remarquer qu'il s'agit d'une complexité très faible, ce qui a été vérifié dans les expérimentations : les suppressions des accords considérés prennent entre 10 et 50 secondes maximum.

Discussion

L'ensemble du processus de construction de l'ontologie globale repose sur l'ontologie de référence, ou plutôt sur sa partie taxonomie. L'utilisation d'une ontologie de référence a déjà été considérée dans les systèmes d'alignement d'ontologies (voir [SdM06, AtKvH06, AKtKvH06, RS07, Ale08]), mais dans le principal but d'augmenter le nombre de mappings découverts entres les concepts des deux ontologies à aligner. Personne à notre connaissance n'en a utilisé pour aller plus loin que l'alignement de deux ontologies et conserver les liens trouvés au sein d'une ontologie globale comme nous le faisons. Même si les ontologies de références ont tendance à se multiplier sur le web sémantique, le choix de la bonne référence pour le domaine qui nécessite la médiation reste un travail en soi, où la question de la langue est présente (une partie des ressources du web sémantique est monolingue - en anglais -, sauf pour les très grands projets tels AGROVOC et SNOMED).

Une fois trouvée la bonne ontologie de référence, pourquoi ne pas l'utiliser directement comme schéma global ? Les ontologies de référence sont souvent de grande taille et peuvent concerner plusieurs domaines connexes. Les utiliser telles quelles comme support d'interrogation pour un système de médiation pose un problème puisqu'une bonne partie des informations modélisées dans l'ontologie de référence peut n'exister dans aucune source. Pour autant, nous ne considérons que la taxonomie de cette ontologie de référence, alors qu'elle peut éventuellement posséder d'autres types de relation et une structure bien plus riche. Mais autant les liens de taxonomie peuvent être étendus naturellement aux concepts subsumés dans les sources sans risque d'inconsistance, autant pour les autres informations présentes dans l'ontologie de référence une étude plus poussée est nécessaire pour déterminer ce qui peut être reproduit dans l'ontologie globale ou pas (selon la relation ou l'attribut, l'étendre à des concepts provenant de sources différentes pourrait ne pas avoir de sens). La taxonomie extraite pour l'instant est nécessaire et suffisante pour l'interrogation.

Le système de construction de schéma médiateur que je viens de présenter a été conçu au départ pour un projet dans lequel plusieurs sources sont d'accord pour partager leurs ressources et décident de collaborer pour construire le système médiateur. Mais son champ d'application est bien plus général : en effet, l'indépendance laissée aux sources que je viens de mettre en évidence dans cette section fait qu'il est possible d'intégrer au système médiateur des sources qui n'ont pas fait le choix d'y participer. Plus précisément, les gestionnaires du médiateur peuvent y ajouter des ressources ouvertes du web sémantique [28] générales comme Geonames [29], ou même DBpedia [30], ou spécifiques au domaine précis de la médiation. Pour cela il suffit d'extraire le schéma de ces ressources (par des requêtes SPARQL) et de l'intégrer au schéma global du médiateur exactement comme pour toute autre source.

Cela permet alors d'interroger *les parties de telles ressources qui sont reliées au domaine de médiation* (sélectionnées grâce à l'ontologie de référence), conjointement avec les autres sources et en pleine conscience du schéma, plutôt que totalement à l'aveuglette comme le proposent la plupart des systèmes actuels d'intégration des données ouvertes liées [QL08, SHH+11, GHS12], dans lesquels l'effort d'intégration repose essentiellement sur l'utilisateur

28. LOD : http ://linkeddata.org/
29. http ://www.geonames.org/
30. http ://dbpedia.org/About

du système. Cette caractéristique extrêmement utile au niveau du web sémantique pourra être testée dans le cadre du projet Personae, qui fait l'objet de la section section suivante.

3.3 Application au projet Personae

Il est un fait très répandu dans les projets d'intégration de ressources dans le domaine des humanités : les équipes partenaires ont d'excellentes raisons pour ne pas abandonner les structures et schémas qu'elles ont élaboré pour les données sur lesquelles elles travaillent. C'est en réalité une des richesses de ces domaines, qui doit être préservée. Le projet Personae est un exemple d'une telle situation. Je présente d'abord le projet et ses objectifs, avant de synthétiser les caractéristiques du médiateur conçu dans ce cadre.

3.3.1 Les grandes lignes du projet Personae

Le projet Personae est porté par le CESR (Centre Etudes Supérieures de la Renaissance) de Tours, financé par la Région Centre sur 2011-2014. Il est le cadre d'une coopération entre chercheurs en histoire et chercheurs en informatique, type de coopération que le CESR met déjà en œuvre depuis quelques années avec succès dans le cadre des BVH [31] (Bibliothèques Virtuelles Humanistes).

Pratique historique traditionnelle qui vise à réaliser des biographies en série de groupes sociaux plus ou moins importants et plus ou moins précisément identifiés (les humanistes de la Renaissance, les professeurs et /ou les étudiants d'une faculté ou d'une université, les artistes, etc.), la prosopographie [LP11] trouve dans les dictionnaires locaux, professionnels ou thématiques, un mode naturel d'expression. Le projet Personae a pour objectif d'élaborer de nouveaux modes d'expression, de publication et de traitements de ces données. Il vise à mettre l'accent sur la reconstitution du tissu humain qui compose les cercles culturels du Moyen-Âge et de la Renaissance et de créer ou de mettre en réseau un ensemble d'outils, de méthodes, de représentations virtuelles ou matérielles en rapport avec la prosopographie du Moyen-Âge ou de la Renaissance.

Un certain nombre de corpus numériques étaient déjà identifiés au départ : étudiants passés par l'université de Poitiers à la fin du Moyen-Âge et à la Renaissance, humanistes transmetteurs de textes, médecins médiévaux et humanistes repérés dans la documentation de l'IRHT (environ 12 000 fiches/personnes) ou dans celle du groupe d'Orléans, astronomes et astrologues de la fin du Moyen-Âge (300 personnes). S'y ajoute le programme Prosopographie des Chantres de la Renaissance (PCR [32]), qui renseigne les biographies des chanteurs professionnels des 15e et 16e siècles (environ 5 000 fiches à terme), qui étaient formés dans les cathédrales et grandes collégiales (surtout dans le Nord de la France et les anciens Pays-Bas) puis exerçaient leurs talents dans les églises et les cours de l'Europe entière. Il y a également des inventaires déjà établis mais pas encore exploitables au format numérique, comme les fichiers manuels conservés dans les institutions partenaires ou dans d'autres institutions : le fichier Picot, issu des dépouillements faits par Emile Picot, donne des listes d'auteurs, de textes et les bibliographies les concernant ; le fichier Lesellier

31. http ://www.bvh.univ-tours.fr/
32. http ://ricercar.cesr.univ-tours.fr/3-programmes/PCR/

dresse un inventaire des Français en rapport avec la papauté à partir du dépouillement des registres de lettres ou de suppliques. Il est prévu par la suite d'élargir les inventaires en intégrant progressivement des champs d'activités différents (naturalistes, peintres, juristes etc.), en partant de la dimension régionale : avec trois universités fondées aux 14e et 15e siècles (Orléans en 1306, Poitiers en 1431, Bourges en 1464), la France du Centre-Ouest est à la pointe de l'innovation à la fin du Moyen-Âge, et d'autres lieux du savoir vont progressivement se mettre en place au cours du 16e siècle.

Ainsi techniquement, plusieurs bases de données existent déjà. L'idée initiale énoncée dans le dossier du projet était de "procéder à une normalisation des métadonnées permettant la description des carrières des personnages étudiés, et à mettre en œuvre les procédures d'exportation des données vers des entrepôts moissonnables à partir d'un portail central". Par "normalisation des méta-données" les partenaires historiens entendaient la constitution d'un schéma unique de base données, a priori sur la base de celui de la plus avancée des bases existantes, la base Budé [33]. L'essentiel des réunions de travail des dix premiers mois a porté là-dessus, chaque choix précis de représentation fait pour Budé, concernant les noms des personnes, les dates, les lieux, etc. a été savamment (longuement et itérativement) discuté, quant à sa pertinence pour les données des autres sources. Petit à petit cependant, j'ai pu faire adopter le principe de laisser à chaque source son mode de représentation logique pour travailler plutôt sur une représentation sémantique commune à mettre en oeuvre sous la forme d'un médiateur (au passage, le principe d'un médiateur plutôt que l'entrepôt initialement prévu, s'est également imposée).

Le projet Personae ne se limite pas à l'intégration de nombreuses sources de données prosopographiques, celle-ci représente plutôt une étape dont le résultat doit permettre d'élaborer de nouveaux modes de publication et d'exploitation de ces données. Par exemple il est prévu une prise en compte des images, et en particulier, la prise en compte des mains [34], qui impose que soit développés des outils spécifiques de comparaison des images et de repérage des similarités pour offrir des possibilités très neuves de description des écritures anciennes. Appliqué à des portraits, ces outils permettraient également de retrouver les représentations successives d'un personnage. Dans un autre registre, les bases de données prosopographiques, quelles que soient leurs spécificités, font état des déplacements des personnages étudiés au fil de leur carrière. Pour cette raison, la visualisation cartographique des données pourrait offrir des représentations renouvelées des lieux de formation, des cercles d'érudition, des milieux de production et de transmission des savoirs. Un des aspects marquants des biographies de musiciens, par exemple, réside dans leur incessante mobilité. Comme les humanistes (et d'autres catégories d'artistes ou de professions intellectuelles), l'élite de ces voyageurs vit dans un espace de dimension européenne, qu'ils parcourent à un rythme étonnant. À cet égard, les interfaces de géolocalisation sont un enjeu déterminant de ce projet : les individus se meuvent dans un espace ; en retour, leurs déplacements créent des réseaux qui modèlent cet espace. Il est prévu de coupler les données intéressant la chronologie et la géographie avec les outils à l'œuvre dans les Système d'Information Géographiques. Grâce à l'adoption progressive par les partenaires du principe d'utiliser des méthodes et standards du web sémantique pour concevoir le portail, ces outils de visualisation et de navigation dans l'espace et le temps utiliseront les données qu'aura

33. http ://bude.irht.cnrs.fr/, voir aussi la communication en ligne : http ://heloise.hypotheses.org/85
34. Ecriture propre à une personne.

intégrées le médiateur et qu'il fournira à travers des API de type points d'accès SPARQL.

3.3.2 Le médiateur de Personae

Le projet Personae dispose de financements de postdoc, en histoire mais également en informatique, pour au moins dix-huit mois. Ce n'est donc pas exactement le cadre d'application de notre proposition, dans lequel des sources veulent partager des données sans avoir les moyens en expertise humaine pour le faire. Samir Sebahi, engagé en postdoc sur ce projet, a plus que l'expertise nécessaire pour construire un médiateur sémantique opérationnel en collaboration avec les chercheurs du CESR. L'intérêt de ce projet est donc surtout de permettre de confronter ce qui peut être construit par notre proposition avec ce qui l'est réellement à partir d'un ensemble d'expertises et de décisions humaines. C'est pourquoi nous menons en parallèle les deux réalisations : celle qui expérimente notre proposition et celle des outils destinés aux chercheurs du CESR, la seconde privilégiant les solutions (libres) immédiatement opérationnelles et bien maintenues.

La figure 3.9 représente le schéma du médiateur Personae. Les différentes sources doivent fournir un schéma conceptuel à partir duquel nous construisons une ontologie légère et, avec le gestionnaire de la source, les mappings d'un système OBDA. De cette manière, une application peut interroger la source à partir de son ontologie, en SPARQL. Si elle le souhaite la source peut publier sur le web son schéma pour faciliter son interrogation par tout public, ou simplement le transmettre aux partenaires auxquels elle souhaite effectivement fournir ses données. Indépendamment de cet accès ouvert aux applications, la source peut publier ses données de la manière qu'elle souhaite via un site web classique. C'est le cas pour la base des Chantres, financée par un programme dans lequel le CESR est engagé à publier son contenu via un site précis. C'est également le cas de la base Budé, qui sert déjà à de nombreux partenaires via des formulaires d'interrogation.

Le système médiateur de Personae intègre les différentes sources selon les principes décrits dans les sections précédentes. A terme il s'appuiera sur une ontologie de référence pour disposer du caractère dynamique évolutif de notre proposition. Nous étudions pour cela les taxonomies de DBPedia d'une part et de YAGO2 d'autre part. Des expérimentations ont été menées avec WOLF [35], version française de Wordnet, qui s'avère être trop généraliste pour fournir un schéma global intéressant. Une référence plus en rapport avec la prosopographie apportera une structure plus riche et le fait de devoir dupliquer en anglais les noms des concepts, rôles et attributs des ontologies sources, tâche aisée, aura en plus pour avantage un couplage simple avec d'autres ressources du web sémantique. L'ensemble des outils de visualisation et de navigation prévus passent par l'interface du médiateur pour exploiter les différentes sources, pour cela ces applications travaillent avec des données en format RDF/OWL. Par ailleurs, des ressources telles que des dictionnaires de prénoms, de métiers, de positions ecclésiastiques ou militaires, etc. doivent être ajoutées au portail pour en enrichir le potentiel de recherche d'information (des pré et post traitements des requêtes permettront d'enrichir les requêtes ou au contraire de préciser les réponses).

Concernant l'interrogation en elle-même, le premier prototype implémenté pour valider les principes auprès de nos collègues historiens s'appuie sur un outil existant d'interrogation

35. http ://alpage.inria.fr/ sagot/wolf-en.html.

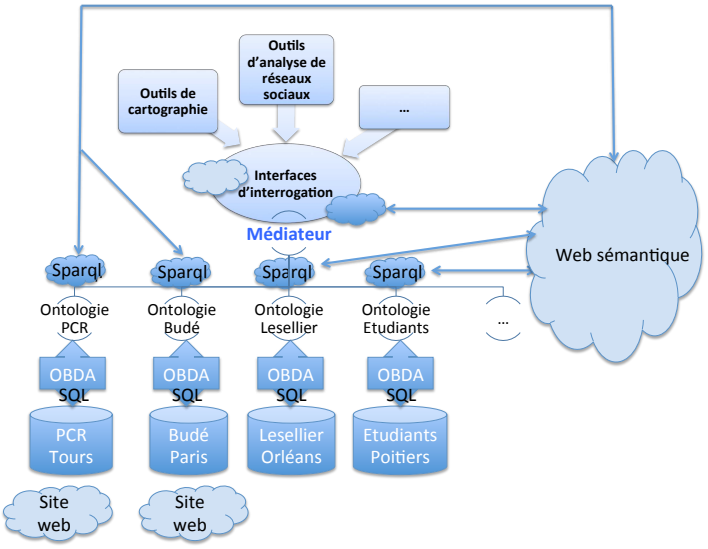


FIGURE 3.9 – Médiateur de Personae

d'une fédération de sources de données liées, Fedx[36] [SHH+11] . La mise au point de l'interrogation décrite dans les sections précédentes est en cours de réalisation, en particulier les dernières étapes, le calcul de la réponse à partir des n-uplets de valeurs retournés par les sources.

3.4 Conclusion et perspectives

L'architecture du schéma global du médiateur élaborée au cours de la thèse de Cheikh Niang peut être comparée aux propositions concernant des *logiques de descriptions distribuées* (DDL) [BS03, SBT05, ST05, GST07, HS10]. Le principe de ces propositions, déjà établi dans [CL93], est que des bases de connaissances distinctes peuvent être reliées par des "ponts" (*bridge rules*) qui consistent en des relations définies entre items de deux bases. De même que les mappings dans [CL01], ces relations peuvent être des subsomptions, dans un sens ou dans l'autre, ou des équivalences entre concepts. Ainsi dans [CL93] sont définis des liens via la relation de *subsomption entre concepts* de deux bases. Plus généralement, dans [BS03] une *TBox distribuée* est définie comme *(i)* une collection de TBoxes T_i représentant les bases locales et *(ii)* des ensembles de règles (ponts) $\mathcal{B}_{i,j}$ qui définissent des relations de subsomption entre concepts des sources S_i et S_j, dans lesquelles est précisée la direction de "transfert" des informations (de la source S_i vers la source S_j ou le contraire). Cette formalisation est étendue dans [GST07] par des relations entre *propriétés* également. Dans [HS10], différentes sémantiques de ces liens sont étudiées, chacune conservant l'as-

36. http ://www.fluidops.com/fedx/.

pect "faiblement couplé" de tels systèmes distribués, dans le sens où les sources restent autonomes, pouvant mettre chacune en oeuvre des mécanismes de raisonnements internes différents. Dans ces propositions le système distribué est de type pair-à-pair. Il en est de même dans [AGR09], où les auteurs réécrivent un système pair-à-pair de TBoxes, chacune exprimée en $DL\text{-}Lite_{\mathcal{R}}$ [37], vers une logique propositionnelle distribuée, déjà implémentée dans leur plateforme SomeWhere [ACG+06].

D'autres mécanismes pour relier des ontologies ont été proposés, par exemple les $\mathcal{E}$-connections suggérées dans [GPS06] : les liens entre les ontologies exprimées en OWL-DL prennent la forme de propriétés, c'est-à-dire en logique de description de rôles reliant un concept de l'ontologie source avec un concept de l'ontologie cible. S'il est impossible de spécifier qu'un tel rôle, appelé "propriété de lien", est transitif ou symétrique, il est cependant possible de leur appliquer des contraintes existant en OWL pour définir de nouveaux concepts non atomiques. L'objectif de ces $\mathcal{E}$-connections est moins de construire un système d'intégration que de *réutiliser* des parties d'ontologies existantes pour la construction d'une nouvelle ontologie. Cet objectif est également traité dans [GR11], où les auteurs définissent la notion de "modules" d'ontologies $DL\text{-}Lite$ et montrent les limites existantes pour une combinaison saine de tels modules.

Comme dans les systèmes de logiques de description distribués, nous relions les ontologies locales via des relations entre leurs concepts et d'autres, appartenant à la taxonomie globale. La principale différence réside dans l'aspect centralisé du médiateur, par rapport à une architecture pair-à-pair. La comparaison avec les systèmes de logiques de description distribués ouvre des perspectives immédiates à notre proposition : nous n'avons étudié que le cas où les liens s'expriment par une relation de subsomption entre des concepts sources et des concepts de la taxonomie globale, il reste donc à explorer *(i)* le cas de relations *d'équivalence plutôt que de subsomption* et (ii) le cas de relations (de subsomption ou d'équivalence) entre *propriétés*, ce qui revient à exploiter d'autres connaissances de l'ontologie de référence que seulement sa partie taxonomie.

Le travail en cours sur le projet Personae ouvre également des perspectives importantes. Par exemple, les questions d'identification de similarités y sont très récurrentes : détecter que deux personnages décrits dans deux bases sont peut être les mêmes [38], en comparant par exemple leurs parcours, leurs maîtres, leur cercle de connaissance, etc. représente un des objectifs fondamentaux des historiens. Il est important d'étudier dans quelle mesure l'architecture du médiateur pourrait faciliter des *définitions de mesures de similarité* utiles pour assister efficacement les recherches des historiens dans les différentes sources reliées par le médiateur.

Un autre besoin mis en évidence par le projet Personae et illustré en figure 3.9 consiste à interconnecter le médiateur à d'autres ressources du web sémantique. Nous avons comme première perspective importante de déterminer si l'intuition exposée en fin de section 3.2, à savoir l'intégration dynamique de parties de ressources ouvertes au sein du médiateur, au même titre que des sources locales privées, est viable. Cependant, il sera aussi important d'étudier l'autre approche, qui consiste à faire du médiateur un pair qui interagit avec ces

37. Les liens entre les pairs sont ici des inclusions (subsomptions), positives ou négatives.

38. A la fin du Moyen Age et à la Renaissance, une personne est couramment mentionnée avec des noms différents selon où elle se trouve et la période de sa vie.

ressources ouvertes, dans un système de type pair-à-pair. Dans un cas comme dans l'autre, en considérant que se développent des catalogues de ressources ontologiques ouvertes, une idée est de repérer dans le schéma global des besoins en ontologies "compagnons", par exemple :

- à partir du moment où sont traitées des notions de temporalité, relier le médiateur à une ontologie dédiée à la description de mesures et d'inférences temporelles,
- à partir du moment où sont traitées des notions de localisation, lui associer un système de connaissances géospatiales,
- à partir du moment où il y a des noms propres, lui associer un système de connaissances culturelles, sociétales,
- de même pour toutes les connaissances scientifiques : biologiques, chimiques, physiques, etc. mais aussi géographiques et historiques, sociologiques, etc.

Un tel repérage pourrait se faire en analysant les labels, les annotations, éventuellement aussi les requêtes posées sur le schéma global. Cette perspective présente l'intérêt d'un travail commun avec mes collègues spécialistes des langues naturelles, alors que nous n'avons pas abordé pour l'instant cet aspect (le rôle du langage naturel) pourtant d'importance capitale pour le web sémantique.

Une autre perspective découle de l'analyse de deux approches complémentaires, *(i)* celle des systèmes OBDA qui en quelque sorte revient à publier une base de données relationnelle "sur" le web sémantique, dans le sens où elle permet un accès aux données de la base relationnelle via le niveau sémantique (les données sources pouvant être enrichies par les connaissances représentées au niveau sémantique), et *(ii)* celle qui consiste à exporter des bases de données relationnelles "dans" le web sémantique avec des mappings décrits en R2RML [39]. Dans la deuxième approche, les données produites sont "dans" le web sémantique dans le sens où elles sont référençables dans ce niveau et peuvent aussi référencer d'autres données sémantiques. Le web sémantique est le web des données liées, or tant que les données sont accessibles seulement via des API, ou des points SPARQL, elles ne sont pas des données liées. Il apparaît que les données d'un système OBDA (gérées dans un SGBD relationnel et accédées via des requêtes sur l'ontologie qui les représente) pourraient être virtuellement "dans" le web des données si elles étaient identifiées via les éléments de l'ontologie, de façon à être référençables (et "déréferençables", c'est-à-dire aisément accessibles depuis le web des données, via un point d'accès SPARQL). Pour définir de telles références (URI) "virtuelles", les principes décrits dans [GKK+13] qui mêlent données XML et données RDF, travail que j'ai déjà évoqué dans les perspectives du chapitre précédent, pourraient être revisités.

39. Standard de description de mappings entre bases de données relationnelles et web sémantique : http ://www.w3.org/TR/r2rml/

Bilan et perspectives

Ce document présente mes contributions à la direction de recherches, réalisées dans l'axe initié dans la première moitié des années 2000, "Services et données du web", de l'équipe "Bases de données et traitement automatique des langues naturelles" (BdTln) du Laboratoire d'Informatique de Tours (EA 6300). Pour mon intégration dans cette équipe, j'avais auparavant travaillé avec Denis Maurel sur la problématique de l'interrogation de bases de données en langue naturelle, laquelle bénéficie d'un regain d'intérêt dans le cadre du web sémantique et des "big data", bien que ce soit sous la forme restreinte de l'interrogation par mots-clés [40]. Comme le montre le chapitre 2 de ce document, j'ai continué à travailler avec l'axe "Traitement automatique des langues naturelles", sur la définition de formats XML pour l'interopérabilité des ressources lexicales multilingues. J'ai également co-encadré avec Denis Maurel entre octobre 2003 et novembre 2007 la thèse de Lamia Tounsi [Tou07], sur la compression d'automates à états finis représentant des dictionnaires électroniques. Les machines à états finis, qui tiennent une place importante dans mes travaux sur XML, étaient déjà au cœur de mon travail de thèse, soutenu en 1995 à l'Université de Franche-Comté (Besançon), avec les machines de réduction paresseuse de graphes de "super-combinateurs", une des formes possibles d'implémentation d'un langage fonctionnel [41].

Des 5 docteurs dont j'ai co-encadré les travaux, 4 sont restés engagés dans la recherche : Denio Duarte (co-encadrement 40%) est enseignant chercheur permanent à l'Université Fédérale de FRONTEIRA SUL (Brésil), Adriana Vidigal de Lima (co-encadrement 45%) est enseignant chercheur permanent à l'Université Fédérale de UBERLÂNDIA (Brésil), Lamia Tounsi (co-encadrement 50%) s'est intégrée dans l'équipe de traitement des langues à l'Université de la ville de Dublin (Irlande) et Cheikh Niang (co-encadrement 75%) est pour l'instant ATER et se destine passionnément à la carrière d'enseignant chercheur. Ahmed Cheriat (co-encadrement 45%) a immédiatement travaillé dans le privé à l'issue de sa thèse pour subvenir aux besoins de sa famille et malgré des tentatives pour revenir vers la recherche il y est finalement resté [42].

Les travaux présentés dans ce document ont été publiés en articles longs de revues : le chapitre 1 synthétise des résultats publiés dans deux articles, [ABS13] pour la correction de documents et [BFL12] pour les contraintes d'intégrité. Le chapitre 2 s'appuie également sur

40. Voir par exemple ce projet européen COST en cours de démarrage : KEYSTONE (http://www.cost.eu/domains_actions/ict/Actions/IC1302).

41. J'y proposais une machine de réduction paresseuse de graphes de super-combinateurs *parallèle et distribuée* pour implémenter le langage fonctionnel parallèle KAP^{2}L.

42. Il est responsable R&D informatique pour le Groupe Kalidea (http://www.groupekalidea.com) et a récemment fait appel aux compétences de l'équipe BdTln en fouille de données.

des articles en journaux ou ouvrages collectifs [BM08, MB13] et est l'objet d'un article de revue encore en cours de rédaction. Le chapitre 3 fait l'objet d'un article soumis fin juillet 2013 au numéro spécial de la revue Journal of Web Semantics sur le thème Ontology-Based Data Access.

Mes travaux à venir aborderont les perspectives énoncées en fin de chacun des chapitres de ce document, ainsi par exemple sur les préférences XML exprimées en requêtes-arbres j'ai proposé un sujet à Maurice Tchoupé, enseignant chercheur camerounais du LIRIMA [43], pour un postdoc de 6 mois. L'orientation de l'axe "Services et données du web" vers le web sémantique a suivi naturellement celle du web, motivée par l'amélioration de l'interopérabilité des applications et des données. Cette même motivation est à l'origine d'un sujet de thèse conçu de longue date avec Yacine Sam, sur le "data exchange" ou la conversion automatique de données d'un schéma à l'autre, pour lequel je n'ai pas encore pu trouver de financement.

Le web sémantique est désormais un domaine de recherche en pleine ébullition, dans lequel beaucoup est à faire sur les aspects formels, où des problématiques classiques en bases de données et en intelligence artificielle sont revisitées avec le point de vue nouveau de la distribution, à très grande échelle. Il soulève également quantité de questions en lien direct avec le TAL, qui sont encore à peine étudiées pour l'instant, j'ai déjà cité l'interrogation en langue naturelle ou ne serait-ce que par mots clés, mais il y a aussi le multilinguisme, élément crucial pour lequel des groupes de réflexion et des workshops spécifiques émergent seulement maintenant [44]. Le travail mené sur Prolexbase et en particulier les choix faits pour le multilinguisme, ainsi que ceux réalisés par les groupes de travail sur LMF et TMF [FBG+09], peuvent éclairer les réflexions en cours dans cette nouvelle communauté. Pour orienter mes perspectives sur ce point j'ai confié à Jianyang LI, étudiant de Master 1 désireux de s'initier à la recherche par un stage hors cursus, l'objectif d'un état de l'art sur la question de la représentation du multilinguisme dans les ontologies et plus généralement dans le web sémantique.

Toujours en lien avec l'axe TALN, concernant les perspectives pour la ressource lexicale Prolmf, de mon point de vue l'approche citée en fin de chapitre 2, avec ses langages XR et XRQ [GKK+12, GKK+13] grâce auxquels XML et RDF sont interconnectés de sorte que chaque données XML soit potentiellement une ressource RDF, est celle qui permettrait le plus naturellement d'intégrer Prolmf "dans" le web sémantique, tout en gardant les avantages du format LMF-XML (par exemple le format XML permet toute manipulation souhaitée sur la structure des résultats des requêtes). Une telle approche ne suppose aucune nouvelle *conversion* d'un format à un autre, ni dans un sens ni dans l'autre : les deux sont gérés séparément mais *interrogés conjointement*. Sur ce point cette approche est supérieure à celles qui consistent à ajouter des annotations RDF (en microformats ou en

43. Laboratoire International pour la Recherche en Informatique et Mathématiques Appliquées, INRIA, regroupant des équipes de recherche de l'Afrique Sub-Saharienne et du Maghreb.

44. Voir par exemple le groupe http://www.multilingualweb.eu/ et la session "machine session" du workshop de 2013 http://www.multilingualweb.eu/rome-report. Egalement le groupe créé par le W3C en avril 2013 : Best Practices for Multilingual Linked Open Data Community Group http://www.w3.org/community/bpmlod/ avec cette réflexion : "If one looks at the various Linked Data (or Semantic Web) related mailing lists, discussion fora, workshops, etc, multilingual or multicultural issues are almost never discussed. URI-s are defined in English, most of the vocabularies we use are documented in only one language ; they may be hard to use for non-English speakers."

RDFa) dans des documents XHTML ou XML, sans rien offrir pour interroger les données dans les deux types de formats conjointement. Dans cette perspective, une première étape relativement simple est d'exprimer en OWL l'ontologie déjà définie lors de la conception de Prolexbase [TGM04, VKM07], pour ensuite travailler sur son couplage avec la ressource Prolmf.

Dans la dernière partie de ce document j'ai montré à travers le projet Personae le potentiel des propositions conçues dans le cadre du web sémantique pour le développement d'environnements pour la recherche en Histoire. Plusieurs autres projets portés par des collègues historiens sont en cours de définition, au cours desquels nos propositions pourront être adaptées et améliorées comme décrit en fin de chapitre 3, notamment le projet de projet européen Biblimos [45] pour lequel j'accueille Mohamedade Farouk Nanne, enseignant chercheur au Département Mathématique et Informatique de l'Université de Nouakchott, de juin à octobre 2013. Le CESR, porteur du grand projet "Intelligence des Patrimoines" [46], définit aussi actuellement plusieurs directions pour lesquelles une collaboration avec nous a un sens, permettant une continuation directe de nos travaux actuels. Par exemple, l'intérêt des systèmes OBDA pour enrichir les requêtes posées sur des bases de données par des connaissances sur le domaine cible, de manière à obtenir des réponses plus complètes, également pour intégrer des données de façon transparente en laissant leur autonomie aux sources, est désormais reconnu à la fois par la communauté des bases de données et par la celle de l'intelligence artificielle. Les contenus des conférences PODS ou IJCAI en 2013 le démontrent, ou encore, en France, le démarrage du projet ANR PAGODE [47], associant le LRI, le LIRMM et le LIG. Ce projet adresse deux aspects fondamentaux des systèmes OBDA, *i)* le passage à l'échelle des techniques d'interrogation des systèmes OBDA, par l'étude des frontières du pouvoir d'expression des requêtes par rapport à leur calculabilité et la mise au point d'algorithmes de résolution efficaces, et *ii)* le traitement des inconsistances des données sources par rapport aux assertions de l'ontologie, soit en assistant la réparation des données, soit par la définition de sémantiques qui tolèrent les inconsistances. Il existe cependant d'autres problématiques contribuant à la généralisation de l'utilisation de systèmes OBDA dans les systèmes d'information à venir, en particulier l'aspect *conception et maintenance* de la partie ontologie (et mappings) de ces systèmes, qui présente des spécificités par rapport à la problématique générale de conception d'ontologies. Ces particularités peuvent permettre une certaine automatisation. Les techniques de rétro-conception de BD relationnelles, l'utilisation d'ontologies de référence et nos propositions élaborées pour l'interrogation de BD relationnelles en langue naturelle sont autant de sources d'inspiration pour la mise au point d'un système de génération d'un système OBDA à partir d'une base de données relationnelle.

45. Porté par Sophie Caratini, anthropologue et spécialiste de la Mauritanie, directrice de recherche CNRS et Francesco Correale, historien spécialiste De l'Afrique du Nord, Labo CITERES - Université de Tours. L'objectif est la constitution d'une bibliothèque virtuelle des sources écrites originales de l'Ouest Saharien.

46. Elaboré à l'origine comme projet de Labex.

47. http://pagoda.lri.fr/

Bibliographie

[ABH+04] M. A. Abrão, B. Bouchou, M. Halfeld Ferrari, D. Laurent, and M. A. Musicante. Incremental constraint checking for XML documents. In *XSym*, number 3186 in LNCS, 2004.

[ABS00] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web - From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, 2000.

[ABS13] Joshua Amavi, Béatrice Bouchou, and Agata Savary. On Correcting XML Documents With Respect to a Schema. *The Computer Journal*, first published online February 14, 2013 ; doi : 10.1093/comjnl/bxt006, 2013.

[ACG+06] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Distributed reasoning in a peer-to-peer setting : Application to the semantic web. *J. Artif. Intell. Res. (JAIR)*, 25 :269–314, 2006.

[ADDWN12] Helen Aristar-Dry, Sebastian Drude, Menzo Windhouwer, and Irina Nevskaya. Rendering endangered lexicons interoperable through standards harmonization : the RELISH project. In *Eight International Conference on Language Resources and Evaluation (LREC 2012)*, 2012.

[ADN+13] Marcelo Arenas, Jonny Daenen, Frank Neven, Martin Ugarte, Jan Van den Bussche, and Stijn Vansummeren. Discovering xsd keys from xml data. In Kenneth A. Ross, Divesh Srivastava, and Dimitris Papadias, editors, *SIGMOD Conference*, pages 61–72. ACM, 2013.

[AFL02a] M. Arenas, W. Fan, and L. Libkin. On verifying consistency of XML specifications. In *ACM Symposium on Principles of Database System*, 2002.

[AFL02b] M. Arenas, W. Fan, and L. Libkin. What's hard about XML schema constraints ? In Springer, editor, *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, number 2453 in LNCS - Lecture Notes in Computer Science, pages 269–278, 2002.

[AGM+06] C. Agafonov, T. Grass, D. Maurel, N. Rossi-Gensane, and A. Savary. La traduction multilingue des noms propres dans PROLEX. *META*, 51/4 :622–636, 2006.

[AGR09] Nada Abdallah, François Goasdoué, and Marie-Christine Rousset. *DL-Lite*$_{\mathcal{R}}$ in the Light of Propositional Logic for Decentralized Data Management. In *IJCAI*, pages 2010–2015, 2009.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Publishing Company, 1995.

[AKtKvH06] Zharko Aleksovski, Michel C. A. Klein, Warner ten Kate, and Frank van Harmelen. Matching Unstructured Vocabularies Using a Background Ontology. In *EKAW*, pages 182–197, 2006.

[AL02] M. Arenas and L. Libkin. A normal form for XML documents. In *ACM Symposium on Principles of Database System*, 2002.

[AL04] Marcelo Arenas and Leonid Libkin. A normal form for XML documents. *ACM Trans. Database Syst.*, 29(1) :195–232, March 2004.

[Ale08] Zharko Aleksovski. *Using Background Knowledge in Ontology Matching*. Phd dissertation,, Vrije U. Amsterdam, June 2008.

[AM86] P. Atzeni and N. M. Morfuni. Functional Dependencies and Constraints on Null Values in Database Relations. *Information and Control*, 70(1), 1986.

[AMR+12] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, and Pierre Senellart. *Web Data Management*. Cambridge University Press, 2012.

[Are06] Marcelo Arenas. Normalization theory for XML. *SIGMOD Record*, 35(4) :57–64, 2006.

[Are09] Marcelo Arenas. XML Integrity Constraints. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 3592–3597. Springer US, 2009.

[ASU88] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers : principles, techniques, and tools*. Addison-Wesley, 1988.

[ASV01] S. Abiteboul, L. Segoufin, and V. Vianu. Representing and Querying XML with incomplete information. In *ACM Symposium on Principles of Database System*, 2001.

[AtKvH06] Zharko Aleksovski, Warner ten Kate, and Frank van Harmelen. Exploiting the Structure of Background Knowledge Used in Ontology Matching. In *Ontology Matching*, 2006.

[AV85] S. Abiteboul and V. Vianu. Transactions and Integrity Constraints. In *ACM Symposium on Principles of Database System*, 1985.

[AvE82] K. R. Apt and M. H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, 29(3), July 1982.

[Bac07] Bruno Bachimont. *Ingénierie des connaissances et des contenus*. Hermès Lavoisier, 2007.

[BAL09] B. Bouchou, M. Halfeld Ferrari Alves, and A. Lima. Contraintes d'intégrité pour XML : visite guidée par une syntaxe homogène. *RSTI - TSI (Technique et Science Informatique)*, 28(3) :331–364, 2009.

[BAM03] B. Bouchou, M. Halfeld Ferrari Alves, and M. Musicante. Tree Automata to Verify Key Constraints. In *Web and Databases (WebDB)*, 2003.

[Bar95] Barnard D.T. and Clarke G. and Duncan N. Tree-to-tree Correction for Document Trees. Technical Report 95-372, Department of Computing and Information Science, Queen's University, Kingston, Ontario, 1995.

[BB79] C. Beeri and P.A. Bernstein. Computational problems related to the design of normal form relational schema. *ACM Trans. on Database Systems*, 4(1) :30–59, 1979.

[BB03] Alexander Borgida and Ronald J. Brachman. Conceptual modeling with description logics. In *The Description Logic Handbook : Theory, Implementation and Applications*, pages 349–372. Cambridge University Press, 2003.

[BBC+00] Domenico Beneventano, Sonia Bergamaschi, Silvana Castano, Alberto Corni, R. Guidetti, G. Malvezzi, Michele Melchiori, and Maurizio Vincini. Information Integration : The MOMIS Project Demonstration. In *VLDB*, pages 611–614, 2000.

[BBFV05] Michael Benedikt, Angela Bonifati, Sergio Flesca, and Avinash Vyas. Adding updates to XQuery : Semantics, optimization, and static analysis. In *XIME-P*, 2005.

[BBG+02] M. Benedikt, G. Bruns, J. Gibson, R. Kuss, and A. Ng. Automated update management for XML integrity constraints. In *Program Language Technologies for XML (PLANX02)*, 2002.

[BBR11] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm, editors. *Schema Matching and Mapping*. Springer, 2011.

[BCF+07] B. Bouchou, A. Cheriat, M. Halfeld Ferrari, D. Laurent, M. A. Lima, and M. A. Musicante. Efficient Constraint Validation for Updated XML Databases. *Informatica*, 31 :285–309, 2007.

[BCG05] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2) :70–118, 2005.

[BCHFAS06a] Béatrice Bouchou, Ahmed Cheriat, Mirian Halfeld Ferrari Alves, and Agata Savary. Integrating Correction into Incremental Validation. In *Proceeding of BDA 06, Lille, France*, 17–20 October 2006.

[BCHFAS06b] Béatrice Bouchou, Ahmed Cheriat, Mirian Halfeld Ferrari Alves, and Agata Savary. XML Document Correction : Incremental Approach Activated by Schema Validation. In *Proceedings of IDEAS 06, Delhi, India*, pages 228–238. IEEE Computer Society, 11–14 December 2006.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, , and Peter F. Patel-Schneider, editors. *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BD07] Béatrice Bouchou and Denio Duarte. Assisting XML Schema Evolution that Preserves Validity. In *Proceedings of SBBD 07, João Pessoa, Paraíba, Brasil*, pages 270–284. SBC, 15–19 October 2007.

[BDA+04] Béatrice Bouchou, Denio Duarte, Mirian Halfeld Ferrari Alves, Dominique Laurent, and Martin A. Musicante. Schema Evolution for XML : A

Consistency-Preserving Approach. In *Proceedings of MFCS 04, Prague, Czech Republic*, volume 3153 of *Lecture Notes in Computer Science*, pages 876–888. Springer, 22–27 August 2004.

[BDAL03] B. Bouchou, Denio Duarte, M. Halfeld Ferrari Alves, and D. Laurent. Extending Tree Automata to Model XML Validation Under Element and Attribute Constraints. In *5th Internat. Conference on Enterprise Information Systems (ICEIS 2003)*, pages 184–191, 2003.

[BDAM09] B. Bouchou, D. Duarte, M. Halfeld Ferrari Alves, and M. Musicante. Extending an XML Type using Updated Data. In *Services and Business Computing Solutions with XML : Applications for Quality Management and Best Processes*, pages 1–21. IGI Global, 2009.

[BDF+01] P. Buneman, S. Davidson, W. Fan, C. Hara, and W. Tan. Reasoning about keys for XML. In *Proceedings of Database and Programmming Languages*, 2001.

[BDF+03] P. Buneman, S. Davidson, W. Fan, C. Hara, and W. C. Tan. Reasoning about keys for XML. *Information Systems*, 28(8), 2003.

[BDH+08] Daan Broeder, Thierry Declerck, Erhard Hinrichs, Stelios Piperidis, Laurent Romary, Nicoletta Calzolari, and Peter Wittenburg. Foundation of a Component-based Flexible Registry for Language Resources and Technology. In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, 2008.

[BdR04] Utsav Boobna and Michel de Rougemont. Correctors for XML Data. In *Proceedings of XSym 04, Toronto, Canada*, volume 3186 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 29–30 August 2004.

[BFK03] M. Benedikt, W. Fan, and G. M. Kuper. Structural properties of XPath fragments. In *ICDT*, 2003.

[BFL12] B. Bouchou, M. Halfeld Ferrari, and M. A. Vidigal Lima. A Grammarware for the Incremental Validation of Integrity Constraints on XML documents under Multiple Updates. *TLDKS*, VI, LNCS 7600 :167–197, 2012.

[BFSW01] P. Buneman, W. Fan, J. Simeon, and S. Weinstein. Constraints for semistructured data and XML. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 30(1) :47–54, 2001.

[BGM04] Elisa Bertino, Giovanna Guerrini, and Marco Mesiti. A Matching algorithm for measuring the structural similarity between an XML documents and a DTD and its applications. *Information Systems*, 29 :23–46, 2004.

[BGM08] Elisa Bertino, Giovanna Guerrini, and Marco Mesiti. Measuring the structural similarity among XML documents and DTDs. *Journal of Intelligent Information Systems*, 30 :55–92, 2008.

[BH03] B. Bouchou and M. Halfeld Ferrari Alves. Updates and incremental validation of XML documents. In Springer, editor, *The 9th International Symposium on Data Base and Programming Languages (DBPL), in conjunction with VLDB*, number 2921 in LNCS, 2003.

[BHdL11] Béatrice Bouchou, Mirian Halfeld Ferrari, and Maria Adriana Vidigal de Lima. Attribute grammar for XML integrity constraint validation. In *International Conference on Database and Expert Systems Applications DEXA (1)*, pages 94–109, 2011.

[BKM99a] Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 96–103. LAWRENCE ERLBAUM ASSOCIATES LTD, 1999.

[BKM99b] Béatrice Bouchou, Barbara Kaltz, and Denis Maurel. Prédicats logiques / prédicats linguistiques pour la consultation de base de données en langue naturelle. *Revue Informatique et Statistique dans les Sciences Humaines*, 35e année, n°1 à 4 :127–149, 1999.

[BLM01] Béatrice Bouchou, Julien Lerat, and Denis Maurel. L'interrogation de bases de données comme application des classes d'objets. In *TALN, Tours, France*, 2001.

[BM99a] Béatrice Bouchou and Denis Maurel. Natural language database query system. In *NLDB, Klagenfurt, Austria*, 1999.

[BM99b] Béatrice Bouchou and Denis Maurel. Une bibliothèque d'opérateurs linguistiques pour la consultation de base de données en langue naturelle. In *TALN, Cargèse, France*, 1999.

[BM08] B. Bouchou and D. Maurel. Prolexbase et LMF : vers un standard pour les ressources lexicales sur les noms propres. *Traitement automate des langues (TAL)*, 49(1), 2008.

[BMM12] Gerhard Budin, Stefan Majewski, and Karlheinz Mörth. Creating Lexical Resources in TEI P5. A Schema for Multi-purpose Digital Dictionaries. *Journal of the Text Encoding Initiative - Online : http ://j-tei.revues.org/522*, November(3), 2012.

[BMW01] A. Brüggeman-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over non-ranked alphabets. Technical Report HKUST TCSC 2001 05, Hong Kong Univ. of Science and Technology Computer Science Center (available at http ://www.cs.ust.hk/tcsc/RR/2001Â-05.ps.gz), 2001.

[BNV07] Geert Jan Bex, Frank Neven, and Stijn Vansummeren. Inferring xml schema definitions from xml data. In *VLDB*, pages 998–1009, 2007.

[Boy11] Leonid Boytsov. Indexing methods for approximate dictionary searching : Comparative analysis. *ACM Journal of Experimental Algorithmics*, 16(1), 2011.

[BPV04] Andrey Balmin, Yannis Papakonstantinou, and Victor Vianu. Incremental validation of XML documents. *ACM Trans. Database Syst.*, 29(4), 2004.

[BS03] Alex Borgida and Luciano Serafini. Distributed Description Logics : Assimilating Information from Peer Sources. Technical Report Technical Report 0303 :12, Instituto Trentino di Cultura, 2003.

[BTM05] Béatrice Bouchou, Mickael Tran, and Denis Maurel. Towards an XML Representation of Proper Names and their Relationships. In *NLDB*, pages 44–55, 2005.

[BW98a] A. Brüggeman-Klein and D. Wood. One-unambiguous regular languages. *Information and Computation*, 142(2) :182–206, 1998.

[BW98b] A. Brüggeman-Klein and D. Wood. Regular tree languages over non-ranked alphabets. unpublished manuscript, 1998.

[CBH+92] William W Cohen, Alex Borgida, Haym Hirsh, et al. Computing least common subsumers in description logics. In *Proceedings of the national conference on artificial intelligence*, pages 754–754. JOHN WILEY & SONS LTD, 1992.

[CDG+02] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. Available on : *http ://www.grappa.univ-lille3.fr/tata*, 1997 (new version 2002).

[CDZ02] Y. Chen, S. Davidson, and Y. Zheng. XKvalidator : A constraint validator for XML. In *Proceedings of ACM Conference on Information and Knowledge Management*, 2002.

[CE11] Christian Chiarcos and Tomaz Erjavec. OWL/DL formalization of the MULTEXT-East morphosyntactic specifications. In *Linguistic Annotation Workshop*, pages 11–20, 2011.

[CGL+07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics : The *DL-Lite* Family. *J. Autom. Reasoning*, 39(3) :385–429, 2007.

[CGL+08] Diego Calvanese, Giuseppe Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and Marco Ruzzi. Data Integration through *DL-Lite*$_\mathcal{A}$ Ontologies. In Klaus-Dieter Schewe and Bernhard Thalheim, editors, *Semantics in Data and Knowledge Bases*, pages 26–47. Springer-Verlag, Berlin, Heidelberg, 2008.

[CGL+09] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Conceptual modeling for data integration. In *Conceptual Modeling : Foundations and Applications - Essays in Honor of John Mylopoulos, volume 5600 of Lecture Notes in Computer Science*, pages 173–197. Springer, 2009.

[CGL+11] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. *Semantic Web*, 2(1) :43–53, 2011.

[CGL+13] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. *Artif. Intell.*, 195 :335–360, 2013.

[Che06] Ahmed Cheriat. *Une méthode de correction de la structure de documents XML dans le cadre d'une validation incrémentale*. Thèse de doctorat, Université François-Rabelais de Tours, 2006.

[Chi10] C. Chiarcos. Grounding an ontology of linguistic annotations in the Data Category Registry. In *LREC 2010 Workshop on Language Resource and Language Technology Standards (LT&LTS)*, pages 37–40, Valetta, Malta, May 2010.

[CHN+12] Christian Chiarcos, Sebastian Hellmann, Sebastian Nordhoff, Steven Moran, Richard Littauer, Judith Eckle-Kohlerz, Iryna Gurevychyz, Silvana Hartmann, Michael Matuschekz, and Christian M. Meyerz. The Open Linguistics Working Group. In *Eight International Conference on Language Resources and Evaluation (LREC 2012*, 2012.

[CL93] T. Catarci and M. Lenzerini. Representing and using Interschema Knowledge in Cooperative Information Systems. *International Journal on Intelligent and Cooperative Information Systems*, 2(4) :375–398, 1993.

[CL01] Diego Calvanese and Maurizio Lenzerini. A framework for ontology integration. pages 303–316. IOS Press, 2001.

[CLL+06] Diego Calvanese, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. Linking data to ontologies : The description logic *DL-Lite*$_{\mathcal{A}}$. In *In Proc. of OWLED 2006*, 2006.

[CM08] Michel Chein and Marie-Laure Mugnier. *Graph-based Knowledge Representation : Computational Foundations of Conceptual Graphs*. Springer Publishing Company, 2008.

[Cod79] E. F. Codd. Extending the database relational model to capture more meaning. In *ACM SIGMOD International Conference on Management of Data*, 1979.

[CZ00] P. Caron and D. Ziadi. Characterization of Glushkov automata. *Theor. Comput. Sci. (TCS)*, 233(1-2) :75–90, 2000.

[Dat03] C. J. Date. *An Introduction to Database Systems, Eighth Edition*. Addison-Wesley, 2003.

[DC92] M. W. Du. and S. C. Chang. A model and a fast algorithm for multiple errors spelling correction. *Acta Informatica*, 29 :281–302, 1992.

[DG12] Mohamed Maamouri David Graff. Developing lmf-xml bilingual dictionaries for colloquial arabic dialects. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[DLM13] Thierry Declerck, Pirsoka Lendvai, and Karlheinz Mörth. Collaborative Tools : From Wiktionary to LMF, for synchronic and diachronic language data. In Francopoulo [Fra13].

[DLP+07] Eladio Domínguez, Jorge Lloret, Beatriz Pérez, Áurea Rodríguez, Ángel L. Rubio, and María A. Zapata. A survey of UML models to XML schemas transformations. In *Proceedings of the 8th international conference on Web information systems engineering*, WISE'07, pages 184–195, Berlin, Heidelberg, 2007. Springer-Verlag.

[DT05] A. Deutsch and V. Tannen. XML queries and constraints, containment and reformulation. *Theoretical Computer Science*, 336(1), 2005.

[Dua05] Denio Duarte. *Une méthode pour l'évolution de schémas XML préservant la validité des documents*. Thèse de doctorat, Université François-Rabelais de Tours, 2005.

[EKGH+12] Judith Eckle-Kohlerz, Iryna Gurevychyz, Silvana Hartmannz, Michael Matuschekz, and Christian M. Meyerz. UBY-LMF : A Uniform Model for Standardizing Heterogeneous Lexical-Semantic Resources in ISO-LMF. In *Eight International Conference on Language Resources and Evaluation (LREC 2012*, 2012.

[EKGH+13] Judith Eckle-Kohlerz, Iryna Gurevychyz, Silvana Hartmannz, Michael Matuschekz, and Christian M. Meyerz. UBY-LMF : exploring the boundaries of language-independent lexicon models. In Francopoulo [Fra13].

[EKM+12] Chantal Enguehard, Soumana Kané, Mathieu Mangeot, Issouf Modi, and Mamadou Lamine Sanogo. Vers l'informatisation de quelques langues d'Afrique de l'Ouest. In *JEP-TALN-RECITAL 2012, Atelier TALAf 2012 : Traitement Automatique des Langues Africaines*, pages 27–40, 2012.

[EM13] Chantal Enguehard and Mathieu Mangeot. LMF for a selection of African Languages. In Francopoulo [Fra13].

[Fan05] W. Fan. XML contraints : Specification, analysis, and applications (invited talk). In *LAIC*, 2005.

[FBG+09] Gil Francopoulo, Nuria Bel, Monte George, Nicoletta Calzolari, Monica Monachini, Mandy Pet, and Claudia Soria. Multilingual resources for NLP in the lexical markup framework (LMF). *Language Resources and Evaluation*, 43(1) :57–70, 2009.

[FHL+12] Flavio Ferrarotti, Sven Hartmann, Sebastian Link, Mauricio Marín, and Emir Muñoz. Performance Analysis of Algorithms to Reason about XML Keys. In *DEXA (1)*, pages 101–115, 2012.

[FL03] S. Farrar and D.T. Langendoen. Markup and the GOLD ontology. In *EMELD Workshop on Digitizing and Annotating Text and Field Recordings*. Michigan State University, July 2003.

[FM03] C. Fellbaum and G. A. Miller. Morphosemantic links in WordNet. *TAL 44*, 2 :69–80, 2003.

[FM04] Nathalie Friburger and Denis Maurel. Finite-state transducer cascade to extract named entities in texts. *Theoretical Computer Science*, 313 :94–104, 2004.

[FMCP13] Gil Francopoulo, Frédéric Marcoul, David Causse, and Grégory Piparo. Global Atlas : extraction of proper nouns from Wikipedia. In Francopoulo [Fra13].

[Fra13] Gil Francopoulo, editor. *LMF : Lexical Markup Framework, theory and practice*. Hermes Science, 2013.

[FSW01] W. Fan, P. Schwenzer, and K. Wu. Keys with upward wildcards for XML. In *DEXA '01 : Proceedings of the 12th International Conference on Database and Expert Systems Applications*, pages 657–667. Springer-Verlag, 2001.

[FTS00] M. F. Fernandez, W.-C. Tan, and D. Suciu. SilkRoute : Trading between Relations and XML. In *Int. World Wide Web Conf. (WWW)*, 2000.

[GEKH+12] Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. Uby - A Large-Scale Unified Lexical-Semantic Resource. In *European Chapter of the Association for Computational Linguistics (EACL)*, 2012.

[GFZC12] Fabien Gandon, Catherine Faron-Zucker, and Olivier Corby. *Le WEB Sémantique*. Dunod, 2012.

[GGM+02] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. Sweetening ontologies with dolce. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, EKAW '02, pages 166–181, London, UK, UK, 2002. Springer-Verlag.

[GHS12] Luis Galarraga, Katja Hose, and Ralf Schenkel. Partout : A Distributed Engine for Efficient RDF Processing. *CoRR*, abs/1212.5636, 2012.

[GI10] F. Gire and H. Idabal. Regular tree patterns : a uniform formalism for update queries and functional dependencies in XML. In *EDBT/ICDT Workshops*, 2010.

[GKK+12] F. Goasdoué, K. Karanasos, Y. Katsis, J. Leblay, I. Manolescu, and S. Zampetakis. Un modèle hybride xml-rdf pour documents annotés. *Ingéniérie des systèmes d'information, RTSI série ISI*, 17(5) :87 – 111, 2012.

[GKK+13] F. Goasdoué, K. Karanasos, Y. Katsis, J. Leblay, I. Manolescu, and S. Zampetakis. Growing triples on trees : a hybrid XML/RDF model for annotated documents. *VLDB Journal*, pages –, 2013.

[GLL+12] G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, M. Ruzzi, and D. F. Savo. Mastro : A reasoner for effective ontology-based data access. In *Proc. of the OWL Reasoner Evaluation Workshop (ORE 2012), volume 858 of CEUR Electronic Workshop Proceedings, http ://ceur-ws.org*, 2012.

[GLR00] Francois Goasdoue, Veronique Lattes, and Marie-Christine Rousset. The use of CARIN language and algorithms for information integration : The PICSEL system. *International Journal of Cooperative Information Systems*, 9(4) :383–401, 2000.

[GLS+05] D. Goecke, H. Langen, F. Sasaki, A. Witt, and S. Farrar. GOLD and discourse : Domain-and community-specific extensions. In *Proceedings of the E-MELD Workshop on Morphosyntactic Annotation and Terminology : Linguistic Ontologies and Data Categories for Language Resources*, Cambridge, Massachusetts, July 2005.

[GM12] Lise Getoor and Ashwin Machanavajjhala. Entity Resolution : Tutorial. In *VLDB*, 2012.

[GPS06] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Combining OWL ontologies using epsilon-Connections. *J. Web Sem.*, 4(1) :40–59, 2006.

[GR11] François Goasdoué and Marie-Christine Rousset. Robust Module-Based Data Management. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99) :1–14, Dec 2011.

[GRS99] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock : A robust clustering algorithm for categorical attributes. *Data Engineering, International Conference on*, 0 :512, 1999.

[Gru09] Tom Gruber. Ontology. *Encyclopedia of Database Systems*, 2009.

[GST07] Chiara Ghidini, Luciano Serafini, and Sergio Tessaris. On relating heterogeneous elements from different ontologies. In *Proceedings of the 6th international and interdisciplinary conference on Modeling and using context*, CONTEXT'07, pages 234–247, Berlin, Heidelberg, 2007. Springer-Verlag.

[Hal07] M. Halfeld Ferrari Alves. *Les aspects dynamiques de XML et spécification des interfaces de services web avec PEWS*. Habilitation à diriger des recherches, Université François Rabelais de Tours, 2007.

[HFR12] Kais Haddar, Héla Fehri, and Laurent Romary. A prototype for projecting HPSG syntactic lexica towards LMF. *Journal of Language Technology and Computational Linguistics*, 27(1) :21–46, 2012.

[HIST05] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation for large-scale semantic data sharing. *VLDB J.*, 14(1) :68–83, 2005.

[HKL+08] Sven Hartmann, Henning Kohler, Sebastian Link, Thu Trinh, and Jing Wang. On the Notion of an XML Key. In Klaus-Dieter Schewe and Bernhard Thalheim, editors, *Semantics in Data and Knowledge Bases*, volume 4925 of *Lecture Notes in Computer Science*, pages 103–112. Springer Berlin / Heidelberg, 2008.

[HKP+09] P. Hitzler, M. Krotzsch, B. Parsia, P.F. Patel-Schneider, and S. Rudolph. *OWL 2 Web Ontology Language Primer*. W3C Recommendation http ://www.w3.org/TR/owl2-primer/, 2009.

[HL10] Sven Hartmann and Sebastian Link. Numerical constraints on XML data. *Inf. Comput.*, 208(5) :521–544, May 2010.

[HLT10] S. Hartmann, S. Link, and T. Trinh. Solving the implication problem for XML functional dependencies with properties. In *Logic, Language, Information and Computation*, volume 6188 of *Lecture Notes in Computer Science*, pages 161–175. Springer Berlin-Heidelberg, 2010.

[HMS+13] Yoshihiko Hayashi, Monica Monachini, Bora Savas, Claudia Soria, and Nicoletta Calzolari. LMF as a foundation for servicized lexical resources. In Francopoulo [Fra13].

[HMU01] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory Languages and Computation*. Addison-Wesley Publishing Company, second edition, 2001.

[Hos06] M. Hosken. Lexicon Interchange Format - A Description. Technical report, Summer Institute for Linguistics of the Institute for Language Information and Technology, Eastern Michigan University, 2006.

[Hos11] Haruo Hosoya. *Foundations of XML processing.* Cambridge University Press, 2011.

[HQ08] Wei Hu and Yuzhong Qu. Falcon-AO : A practical ontology matching system. *Web Semant.*, 6(3) :237–239, September 2008.

[HS03] S. Hartmann and S.Link. More functional dependencies for XML. In *Advances in Databases and Information Systems - ADBIS*, 2003.

[HS10] Martin Homola and Luciano Serafini. Augmenting Subsumption Propagation in Distributed Description Logics. *Applied Artificial Intelligence*, 24(1-2) :39–76, 2010.

[HSBW13] Johannes Hoffart, Fabian Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2 : A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *Artificial Intelligence Journal*, Special Issue, 2013.

[HSZR09] Fayçal Hamdi, Brigitte Safar, Haïfa Zargayouna, and Chantal Reynaud. Partitionnement d'ontologies pour le passage à l'échelle des techniques d'alignement. In *EGC*, pages 409–420, 2009.

[HT06] S. Hartmann and T. Trinh. Axiomatising functional dependencies for XML with frequencies. In *Foundations of Information and Knowledge Systems (FoIKS), 4th Int. Symposium*, pages 159–178, 2006.

[HZQ06] Wei Hu, Yuanyuan Zhao, and Yuzhong Qu. Partition-Based Block Matching of Large Class Hierarchies. In *ASWC*, pages 72–83, 2006.

[IL84] T. Imieliński and W. Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4) :761–791, 1984.

[Imi89] T. Imielinski. Incomplete information in logical databases. *IEEE Data Engineering Bulletin*, 12(2), 1989.

[ISO03] ISO. *Computer applications in terminology - Terminological markup framework, reference ISO 16642 :2003.* ISO, 2003.

[ISO08] ISO/TC 37/SC 4. *Language resource management - Lexical markup framework (LMF), working document Rev. 16.* http ://www.lexicalmarkupframework.org, 2008.

[ISO09] ISO. *Terminology and other language and content resources ?Specification of data categories and management of a Data Category Registry for language resources, reference ISO 12620 :2009.* ISO, 2009.

[KGHH13] Aida Khemakhem, Bilel Gargouri, Kais Haddar, and Abdelmajid Ben Hamadou. LMF for Arabic. In Francopoulo [Fra13].

[KK03] T. Krumbein and T. Kudrass. Rule-based generation of XML schemas from UML class diagrams. In *WebDB*, pages 213–227, 2003.

[KLV00] George Karakostas, Richard J. Lipton, and Anastasios Viglas. On the complexity of intersecting finite state automata. In *IEEE Conference on Computational Complexity*, 2000.

[Len02] Maurizio Lenzerini. Data integration : A theoretical perspective. In *PODS*, pages 233–246, 2002.

[Lib07] Leonid Libkin. Normalization Theory for XML. In *XSym*, pages 1–13, 2007.

[Lie82] Y. Edmund Lien. On the equivalence of database models. *Journal of the ACM*, 29(2), 1982.

[Lim07] Maria Adriana Vidigal Lima. *Maintenance incrémentale des contraintes d'intégrité en XML*. Thèse de doctorat, Université François-Rabelais de Tours, 2007.

[Lip81] W. Lipski. On databases with incomplete information. *Journal of the ACM*, 28(1) :41–70, 1981.

[LL99] M. Levene and G. Loizou. *A guided tour of relational databases and beyond*. Springer-Verlag, 1999.

[LLL02] M. L. Lee, T. W. Ling, and W. L. Low. Designing functional dependencies for XML. In *EDBT '02 : Proceedings of the 8th International Conference on Extending Database Technology*, pages 124–141. Springer-Verlag, 2002.

[LMCC02] D. Lee, M. Mani, F. Chiu, and W. W. Chu. Net & Cot : Translating relational schemas to XML schemas. In *Australasian Database Conference*, 2002.

[LP11] Claire Lemercier and Emmanuelle Picard. Quelle approche prosopographique ? In *halshs-00521512, version 2*, 2011.

[LRO96] A. Y. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *VLDB*, 1996.

[LTC13] Eric Laporte, Elsa Tolone, and Matthieu Constant. Conversion of Lexicon-Grammar tables to LMF : application to French. In Francopoulo [Fra13].

[LVL03] J. Liu, M. W. Vincent, and C. Liu. Functional dependencies, from relational to XML. In *Ershov Memorial Conference*, pages 531–538, 2003.

[LY08] T. Lv and P. Yan. Removing XML data redundancies by constraint-tree-based functional dependencies. In *CCCM '08 : Proceedings of the 2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, pages 595–599, Washington, DC, USA, 2008. IEEE Computer Society.

[Mai80] D. Maier. Minimum covers in the relational database model. *J. ACM*, 27(4) :664–674, 1980.

[Mau08] D. Maurel. Prolexbase. a multilingual relational lexical database of proper names. In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, 2008.

[MB13] Denis Maurel and Béatrice Bouchou. Prolmf, a multilingual dictionary of proper nouns and their relations. In Francopoulo [Fra13].

[MC96] Marie-Laure Mugnier and Michel Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'Intelligence Artificielle*, 10(1), 1996.

[MFN+12] Denis Maurel, Nathalie Friburger, Damien Nouvel, Iris Eshkol-Taravella, and Jean-Yves Antoine. Cascade de transducteurs : Applications autour des entités nommées. *Traitement Automatique des Langues (TAL)*, 52-1, 2012.

[MGH+09] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. *OWL 2 Web Ontology Language Profiles*. W3C Recommendation http ://www.w3.org/TR/owl2-profiles/, 2009.

[MLM01] M. Murata, D. Lee, and M. Mani. Taxonomy of XML schema language using formal language theory. In *Extreme Markup Language, Montreal, Canada*, 2001.

[MLMK05] Makoto Murata, Dongwon Lee, Murali Mani, and Kohsuke Kawaguchi. Taxonomy of XML schema languages using formal language theory. *ACM Trans. Internet Technol.*, 5(4) :660–704, November 2005.

[MMS79] D. Maier, A.O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Trans. On Database Systems*, 4(4) :455–469, 1979.

[Mon13] George Monte. LMF in U.S. Government Language Resource Management. In Francopoulo [Fra13].

[MPB08] Sanjay Madria, Kalpdrum Passi, and Sourav Bhowmick. An XML Schema integration and query mechanism system. *Data and Knowledge Engeneering*, 65(2) :266–303, May 2008.

[MSC11] J. McCrae, D. Spohr, and P. Cimiano. Linking lexical resources and ontologies on the semantic web with Lemon. *The Semantic Web : Research and Applications*, pages 245–259, 2011.

[MvH04] D. L. McGuinness and F. van Harmelen. *OWL Web Ontology Language Overview*. W3C Recommendation http ://www.w3.org/TR/owl-features/, 2004.

[MVK07] D. Maurel, D. Vitas, and S. Koeva. Prolex : A Lexical Model for Translation of Proper Names. Application to French, Serbian and Bulgarian. *Bulag*, 32 :55–72, 2007.

[MVKK07] D. Maurel, D. Vitas, S. Krstev, and S. Koeva. Prolex : a lexical model for translation of proper names. application to french, serbian and bulgarian. *Bulag*, 32 :55–72, 2007.

[NAFM10] Damien Nouvel, Jean-Yves Antoine, Nathalie Friburger, and Denis Maurel. An Analysis of the Performances of the CasEN Named Entities Recognition System in the Ester2 Evaluation Campaign. In *7th International Language Resources and Evaluation (LREC'10)*, 2010.

[NBL10] Cheikh Niang, Béatrice Bouchou, and Moussa Lo. Towards tailored domain ontologies. In *OM*, 2010.

[NBLS11a] Cheikh Niang, Béatrice Bouchou, Moussa Lo, and Yacine Sam. Appropriate global ontology construction : a domain-reference-ontology based approach. In *iiWAS*, pages 166–173, 2011.

[NBLS11b] Cheikh Niang, Béatrice Bouchou, Moussa Lo, and Yacine Sam. Automatic building of an appropriate global ontology. In *ADBIS*, pages 429–443, 2011.

[NBLS12] Cheikh Niang, Béatrice Bouchou, Moussa Lo, and Yacine Sam. Querying a semi-automated data integration system. In *DEXA (2)*, pages 305–313, 2012.

[NBLS13] Cheikh Niang, Béatrice Bouchou, Moussa Lo, and Yacine Sam. Ré-écriture de requêtes dans un système d'intégration sémantique. In *EGC*, pages –, 2013.

[NBSL13] Cheikh Niang, Béatrice Bouchou, Yacine Sam, and Moussa Lo. A Semi-Automatic approach For Global-Schema Construction in Data Integration Systems. *IJARAS*, 4(2) :35–53, 2013.

[Nev99] F. Neven. Extensions of attribute grammars for structured document queries. In *Proceedings of International Workshop on Database Programming Languages*, 1999.

[Nia13] Cheikh Ahmed Tidiane Niang. *Vers plus d'automatisation dans la construction de systèmes médiateurs pour le web sémantique.* Thèse de doctorat, Université François-Rabelais de Tours, 2013.

[Odi13] Jan Odijk. DUELME : Dutch Electronic Lexicon of Multiword Expressions. In Francopoulo [Fra13].

[Ofl96] K. Oflazer. Error-tolerant Finite-state Recognition with Applications to Morphological Analysis and Spelling Correction. *Computational Linguistics*, 22(1) :73–89, 1996.

[OMG02] OMG. XML Metadata Interchange (XMI) Specification. Technical report, http ://www.omg.org/cgi-bin/doc ?formal/2002-01-01, 2002.

[PH01] Rachel Pottinger and Alon Y. Halevy. Minicon : A scalable algorithm for answering queries using views. *VLDB J.*, 10(2-3) :182–198, 2001.

[PLC+08] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking Data to Ontologies. *J. Data Semantics*, 10 :133–173, 2008.

[PLL+13] Floriana Di Pinto, Domenico Lembo, Maurizio Lenzerini, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Optimizing query rewriting in ontology-based data access. In *EDBT*, pages 561–572, 2013.

[PMC98] Denis Le Pesant and Michel Mathieu-Colas. Introduction aux classes d'objets. *Languages*, 131, 1998.

[PRS12] Gabriele Puppis, Cristian Riveros, and Slawek Staworko. Bounded repairability for regular tree languages. In Alin Deutsch, editor, *ICDT*, pages 155–168. ACM, 2012.

[PV13] Monica Monachini Piek Vossen, Claudia Soria. Wordnet-LMF : a standard representation for multilingual wordnets. In Francopoulo [Fra13].

[QACT12] Anna Queralt, Alessandro Artale, Diego Calvanese, and Ernest Teniente. OCL-Lite : Finite reasoning on UML/OCL conceptual schemas. *Data and Knowledge Engineering*, 73 :1–22, 2012.

[QL08] Bastian Quilitz and Ulf Leser. Querying distributed rdf data sources with sparql. In *ESWC*, pages 524–538, 2008.

[RA10] Riccardo Rosati and Alessandro Almatelli. Improving Query Answering over DL-Lite Ontologies. In *KR*, 2010.

[RMC12] Mariano Rodriguez-Muro and Diego Calvanese. Quest, an OWL 2 QL Reasoner for Ontology-based Data Access. In *OWLED*, 2012.

[Rom10] Laurent Romary. Standardization of the formal representation of lexical information for NLP. In *Dictionaries. An International Encyclopedia of Lexicography. Supplementary volume : Recent developments with special focus on computational lexicography*. Mouton de Gruyter, 2010.

[Rom11] Laurent Romary. Stabilizing knowledge through standards - A perspective for the humanities. In Karl Grandin, editor, *Going Digital : Evolutionary and Revolutionary Aspects of Digitization*. Science History Publications, 2011.

[Rom13] Laurent Romary. TEI and LMF crosswalks. *http ://fr.arXiv.org/abs/1301.2444*, 2013.

[RR04] Marie-Christine Rousset and Chantal Reynaud. Knowledge representation for information integration. *Inf. Syst.*, 29(1) :3–22, 2004.

[RS07] Chantal Reynaud and Brigitte Safar. Exploiting WordNet as Background Knowledge. In *Proceedings of the Workshop on Ontology Matching (OM2007) at ISWC/ASWC2007, Busan, South Korea*, November 2007.

[RTHB12] Giuseppe Rizzo, Raphaël Troncy, Sebastian Hellmann, and Martin Bruemmer. NERD meets NIF : Lifting NLP Extraction Results to the Linked Data Cloud. In *WWW'12 : 5th Workshop on Linked Data on the Web (LDOW'12)*, Lyon, France, April 2012.

[RW12] Laurent Romary and Werner Wegstein. Consistent modelling of heterogeneous lexical structures. *Journal of the Text Encoding Initiative*, 2012.

[SBT05] Luciano Serafini, Alexander Borgida, and Andrei Tamilin. Aspects of Distributed and Modular Ontology Reasoning. In *IJCAI*, pages 570–575, 2005.

[SC06] Slawomir Staworko and Jan Chomicki. Validity-Sensitive Querying of XML Databases. In *Proceedings of EDBT 06, Munich, Germany, Revised Selected Papers*, volume 4254 of *Lecture Notes in Computer Science*, pages 164–177. Springer, 26–31 March 2006.

[Sch05] Klaus-Dieter Schewe. Redundancy, Dependencies and Normal Forms for XML Databases. In *ADC*, pages 7–16, 2005.

[SdM06] Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Using the semantic web as background knowledge for ontology mapping. In *Ontology Matching*, volume 225 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[SE05] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *J. Data Semantics IV*, pages 146–171, 2005.

[SE08] Pavel Shvaiko and Jérôme Euzenat. Ten challenges for ontology matching. In *OTM Conferences (2)*, pages 1164–1182, 2008.

[SE13] Pavel Shvaiko and Jérôme Euzenat. Ontology matching : State of the art and future challenges. *IEEE Trans. Knowl. Data Eng.*, 25(1) :158–176, 2013.

[Sel77] S. M. Selkow. The Tree-to-Tree Editing Problem. *Information Processing Letters*, 6(6) :184–186, 1977.

[SFC08] Slawomir Staworko, Emmanuel Filiot, and Jan Chomicki. Querying Regular Sets of XML Documents. In *Proceedings of LiD 08, Rome, Italy*, 2008.

[SHH+11] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. Fedx : Optimization techniques for federated query processing on linked data. In *International Semantic Web Conference (1)*, pages 601–616, 2011.

[SHS04] G. M. Sur, J. Hammer, and J. Simeon. An XQuery-based language for processing updates in XML. In *PLAN-X - Programming Language Technologies for XML A workshop colocated with POPL 2004*, 2004.

[SL09] Md. S. Shahriar and J. Liu. On the performances of checking XML key and functional dependency satisfactions. In *OTM Conferences (2)*, pages 1254–1271, 2009.

[SM11] Martin Svoboda and Irena Mlýnková. Correction of Invalid XML Documents with Respect to Single Type Tree Grammars. In *Proceedings of NDT 11, Macau, China*, volume 136 of *Communications in Computer and Information Science*, pages 179–194. Springer, 11–13 July 2011.

[SMB13] A. Savary, L. Manicki, and M. Baron. Prolexfeeder - populating a multilingual ontology of proper names from open sources. *XXX*, X(X) :X–X, 2013.

[SMS11] Malgorzata Spedzia, Denis Maurel, and Agata Savary. Multilingual Relational Database of Proper Names : Prolexbase Documentation. Technical Report 297, Université François Rabelais Tours, Laboratoire d'Informatique, 2011.

[ST05] Luciano Serafini and Andrei Tamilin. Drago : Distributed reasoning architecture for the semantic web. In *ESWC*, pages 361–376. Springer, 2005.

[Suz07] Nobutaka Suzuki. Finding K Optimum Edit Scripts between an XML Document and a RegularTree Grammar. In *Proceedings of EROW 07, Barcelona, Spain*. CEUR-WS.org, 13 January 2007.

[Svo10] Martin Svoboda. Processing of Incorrect XML Data. Master's thesis, Charles University in Prague, 2010.

[Sér12] Gilles Sérasset. Dbnary : Wiktionary as a LMF based Multilingual RDF network. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[TCTT11] Joe Tekli, Richard Chbeir, Agma Traina, and Caetano Traina. XML document-grammar comparison : related problems and applications. *Central European Journal of Computer Science*, 1 :117–136, 2011.

[TCY07] Joe Tekli, Richard Chbeir, and Kokou Yétongnon. Structural Similarity Evaluation Between XML Documents and DTDs. In *Proceedings of WISE 07, Nancy, France*, pages 196–211, 3–7 December 2007.

[TGM04] M. Tran, T. Grass, and D. Maurel. An ontology for multilingual treatment of proper names. In *OntoLex 2004, in Association with LREC2004*, pages 75–78, 2004.

[TLS+13] Takenobu Tokunaga, Sophia Y. M. Lee, Virach Sornlertlamvanich, Kiyoaki Shirai, Shu-Kai Hsieh, and Chu-Ren Huang. LMF and Its Implementation in Some Asian Languages. In Francopoulo [Fra13].

[Tou07] Lamia Tounsi. *Sous-automates à nombre fini d'états : application à la compression de dictionnaires électroniques.* Thèse de doctorat, Université François-Rabelais de Tours, 2007.

[Tra06] Mickaël Tran. *Prolexbase. Un dictionnaire relationnel multilingue de noms propres : conception, implémentation et gestion en ligne.* Thèse de doctorat, Université François-Rabelais de Tours, 2006.

[TVY08] Alex Thomo, Srinivasan Venkatesh, and Ying Ying Ye. Visibly Pushdown Transducers for Approximate Validation of Streaming XML. In *Proceedings of FoIKS 08, Pisa, Italy*, volume 4932 of *Lecture Notes in Computer Science*, pages 219–238. Springer, 11–15 February 2008.

[TZ11] Zijing Tan and Liyong Zhang. Repairing XML functional dependency violations. *Inf. Sci.*, 181(23) :5304–5320, 2011.

[Via01] V. Vianu. A web odyssey : from Codd to XML. In *ACM Symposium on Principles of Database System*, 2001.

[VKM07] D. Vitas, S. Krstev, and D. Maurel. A note on the semantic and morphological properties of proper names in the Prolex project. *Lingvisticae Investigationes*, 30(1) :115–133, 2007.

[VKM09] D. Vitas, S. Krstev, and D. Maurel. A note on the semantic and morphological properties of proper names in the Prolex project. In Satoshi Sekine and Elisabete Ranchhod, editors, *Named Entities : Recognition, classification and use*, pages 117–136. John Benjamins publishing company, 2009.

[VL05] M.W. Vincent and J. Liu. Checking functional dependency satisfaction in XML. In *3rd In- ternational Symposium on Database and XML Technologies*, pages 4–17, 2005.

[VLL04] Millist W. Vincent, Jixue Liu, and Chengfei Liu. Strong functional dependencies and their application to normal forms in XML. *ACM Trans. Database Syst.*, 29(3) :445–462, September 2004.

[VLM07] M. W. Vincent, J. Liu, and M. Mohania. On the equivalence between FDs in XML and FDs in relations. *Acta Informatica*, 44, 2007.

[VLM12] M. W. Vincent, J. Liu, and M. Mohania. The implication problem for 'closest node' functional dependencies in complete XML documents. *J. Comput. Syst. Sci.*, 78(4) :1045–1098, July 2012.

[VPB13] Marta Villegas, Muntsa Padró, and Núria Bel. LMF experiments on format conversions for resource merging : converters and problems. In Francopoulo [Fra13].

[Wal12] Priscilla Walmsley. *Definitive XML Schema 2nd edition.* Prentice Hall, 2012.

[WF74] R. A. Wagner and M. J. Fischer. The String-to-String Correction Problem. *Journal of the ACM*, 21(1) :168–173, 1974.

[Win12] M. Windhouwer. RELcat : a relation registry for ISOcat data categories. In *Eight International Conference on Language Resources and Evaluation (LREC 2012)*, 2012.

[WP94] Zhibiao Wu and Martha Stone Palmer. Verb Semantics and Lexical Selection. In *ACL*, pages 133–138, 1994.

[WPN+13] Menzo Windhouwer, Justin Petro, Irina Nevskaya, Sebastian Drude, Helen Aristar-Dry, and Jost Gippert. Creating a serialization of LMF : the experience of the RELISH project. In Francopoulo [Fra13].

[WT05] J. Wang and R. Topor. Removing XML data redundancies using functional and equality- generating dependencies. In *16th Australasian Database Conference*, 2005.

[WVV+01] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-Based Integration of Information - A Survey of Existing Approaches. In *IJCAI'01 Workshop on Ontologies and Informations Sharing*, pages 108–117, 2001.

[WW12] M. Windhouwer and S. E. Wright. Linking to linguistic data categories in ISOcat. In *Linked Data in Linguistics (LDL 2012)*, Frankfurt/M., Germany, Mar 2012.

[XMXV06] Guangming Xing, Chaitanya R. Malla, Zhonghang Xia, and Snigdha Dantala Venkata. Computing Edit Distances Between an XML Document and a Schema and its Application in Document Classification. In *Proceedings of SAC 06, Dijon, France*, pages 831–835. ACM, 23–27 April 2006.

[YJ08] Cong Yu and H. V. Jagadish. XML schema refinement through redundancy detection and normalization. *The VLDB Journal*, 17(2) :203–223, March 2008.

[ZHT12] Claus Zinn, Christina Hoppermann, and Thorsten Trippel. The ISOcat registry reloaded. In *Proceedings of the 9th international conference on The Semantic Web : research and applications*, ESWC'12, pages 285–299, Berlin, Heidelberg, 2012. Springer-Verlag.

[ZXZ09] X. Zhao, J. Xin, and E. Zhang. XML functional dependency and schema normalization. In *HIS '09 : Proceedings of the 9th International Conference on Hybrid Intelligent Systems*, pages 307–312, 2009.

Résumé

Il est ici question de gestion des données du web, c'est-à-dire de *gestion des données* d'une part et d'autre part de définition de *formats et d'infrastructures pour l'interopérabilité*, raisons d'être du web. Ce fil conducteur passe d'abord par XML et ses *modèles logiques de données*, composés de spécifications de *contraintes de structure* et de *contraintes d'intégrité*. Les propositions exposées explorent comment ces spécifications sont formalisables et vérifiables et comment peut être prise en compte l'évolution constante des documents et des schémas. Puis le fil passe par la question de la mise au point de *formats de représentation* de ressources, ici ressources lexicales, qui favorisent l'interopérabilité des applications qui vont les utiliser, et par là, également la *pérennité* de ces ressources. Les propositions formulées à propos des formats de représentation interopérables amènent à la troisième partie, sur le *web sémantique*. Une intégration d'ontologies "légères" y est décrite, ces ontologies étant définies comme des interfaces sémantiques à travers lesquelles publier et interroger les données. Clairement, il s'agit d'une proposition pour *l'interopérabilité sémantique*, après celles pour l'interopérabilité syntaxique développées précédemment. Chacune des trois parties participe donc de l'amélioration de l'interopérabilité, tant au niveau syntaxique que sémantique, ce qui augmente le potentiel du web en termes d'échanges et de collaborations dans l'élaboration de nos connaissances.

Mots clés : XML, Schémas (définition, validation, évolution), Contraintes d'intégrité (définition, validation), Automates et grammaires d'arbres, Conception XML, Ressource lexicale de noms propres Prolexbase, standard LMF, ressource standardisée Prolmf, Système d'intégration sémantique, OBDA (Ontology Based Data Access), Logique $DL\text{-}Lite_{\mathcal{A}}$, Projet Personae.

Abstract

This dissertation is about *web data management*, therefore, on the one hand it deals with *data management*, and, on the other hand, it addresses the definition of *formats and infrastructures for interoperability*, which are the web's raison d'être. This breadcrumb lets start with XML and its ability to define both *structure constraints* and *integrity constraints*, as any other logical level of data representation. In this part, the contributions investigate the ways of formalizing and automatically verifying all these kinds of constraints. Moreover, the evolution of documents and schemas is taken into account within the proposed validation processes. In the second part we turn ourselves to the development of *formats for digital resources*, which are here lexical resources, with the aim of promoting the interoperability of applications, and thus the sustainability of these resources.Then, our thinkings on representation formats for interoperability purposes lead to the third part, which is related to the *Semantic Web*. An *integration system for lightweight ontologies* is specified, each of these lightweight ontologies being defined as a semantic interface for publishing and querying data. Clearly, this is a proposal for *semantic interoperability*, after the previous two that focused on syntactic interoperability. Thus, each of these three parts contributes to interoperability enhancement, at both syntactic and semantic levels, and as such, increases the web's ability to support collaborations for creating and managing our knowledge.

Keywords : XML, Schema (definition, validation, evolution), Integrity constraints (definition, validation), XML Design, Prolexbase lexical resource on proper names, LMF standard, standardized resource Prolmf, Semantic integration system, OBDA (Ontology Based Data Access), $DL\text{-}Lite_{\mathcal{A}}$ logic, Personae project.